HAL
open science

Data Mining : une approche par les graphes

Alain Sigayret

To cite this version:

Alain Sigayret. Data Mining : une approche par les graphes. Sciences de l'ingénieur [physics]. Université Blaise Pascal - Clermont-Ferrand II, 2002. Français. NNT : . tel-00521946

HAL Id: tel-00521946

https://theses.hal.science/tel-00521946v1

Submitted on 29 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université Blaise Pascal - Clermont II

Ecole Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

T h è s e

Présentée par

Alain SIGAYRET

Formation Doctorale :
Informatique, Productique et Imagerie Médicale

pour obtenir le grade de

Docteur d'Université

spécialité : Informatique

Data Mining : une approche par les graphes

Soutenue publiquement le vendredi 20 décembre 2002 devant le jury :

M. Philippe MAHEY	*Président*
M. Jean–Paul BORDAT	*Rapporteur, examinateur*
M. Alain GUÉNOCHE	*Rapporteur, examinateur*
M. Lhouari NOURINE	*Examinateur*
M. Maurice POUZET	*Examinateur*
Mme Anne BERRY	*Co-directrice de thèse*
M. Alain QUILLIOT	*Directeur de thèse*

La dernière démarche de la raison est de reconnaître qu'il y a une infinité de choses qui la dépasse.

Blaise Pascal

La réalisation d'une thèse, bien qu'étant par nature un travail individuel, ne peut se concevoir sans le soutien de tous ceux qui ont contribué à son aboutissement.

Ainsi, je voudrais tout d'abord remercier Philippe Mahey qui a accepté de présider ce jury et a contribué à me donner les moyens de faire une thèse.

Ce que je sais de l'algorithmique, je le dois à Jean-Paul Bordat ; cet apprentissage s'était prolongé durant mon DEA, et au-delà, par des discussions passionnantes sur les ensembles ordonnés. Je suis donc particulièrement heureux qu'il se soit intéressé à cette thèse, non seulement pour en faire un rapport, mais aussi pour suggérer des pistes de recherche intéressantes et accepter une collaboration sur certaines d'entre elles. Je lui renouvelle ici toute ma gratitude.

Au début de cette thèse, Alain Guénoche avait déjà été de bon conseil pour des choix difficiles. Le travail de lecture attentif qu'il a réalisé sur ce mémoire afin d'en faire un rapport m'a aidé à mettre mieux en évidence les points forts de la première partie. Je l'en remercie très chaleureusement.

Je souhaiterais également remercier le professeur Maurice Pouzet qui s'est intéressé à mon travail et en a donné une évaluation très encourageante.

J'avais déjà eu l'occasion de travailler avec Lhouari Nourine pour mon mémoire de DEA à Montpellier ; j'ai eu le plaisir de le retrouver à Clermont-Ferrand dans l'équipe d'Algorithmique du LIMOS et dans ce jury.

Alain Quilliot a dirigé cette thèse avec une grande ouverture d'esprit et m'a intégré dans l'équipe d'Algorithmique du LIMOS. Je le remercie sincèrement, ainsi que David Hill qui a su, en début de thèse, me suggérer une orientation vers des thèmes en rapport avec mes goûts et mes compétences.

Marianne Huchard commençait sa carrière d'enseignante-chercheuse quand je l'ai eu pour enseignante en licence et maîtrise d'informatique ; elle a ensuite suivi mon stage de DEA et m'a encouragé à poursuivre la recherche. Les discussions récentes qu'elle a eues avec nous m'ont permis de progresser. Je lui dois beaucoup et je tiens à la remercier ici.

J'étais certes familiarisé, avant de commencer cette thèse, avec les travaux d'Anne Berry, mais je n'avais pas encore réalisé la diversité de ses thèmes de recherche centrés initialement sur les graphes ni la puissance de son travail et de son intuition. Ces années intenses m'ont permis d'aborder quelques uns de ces thèmes et de proposer quelques avancées. Je lui exprime ici, très sincèrement, toute ma gratitude pour ce qu'elle m'a apporté et pour avoir accepté de poursuivre notre collaboration.

Enfin, j'adresse un remerciement collectif au LIMOS ainsi qu'au personnel administratif de l'ISIMA et de l'UFR Sciences pour le soutien qu'ils m'ont apporté durant ces années.

La réalisation d'une thèse est aussi une période difficile pour l'entourage ; je remercie particulièrement ma compagne qui m'a soutenu sans faiblir durant ces années, ainsi que le jeune et patient Axel.

Table des matières

Introduction **11**

1 Définitions et notations **15**

1.1 Relations binaires 15

1.1.1 Notations ensemblistes 15

1.1.2 Relations 16

1.1.3 Graphes orientés 18

1.2 Ensembles ordonnés 20

1.2.1 Ordres 21

1.2.2 Treillis 25

1.2.3 Fermetures 28

1.2.4 Treillis de Galois, treillis des concepts 29

1.3 Graphes non-orientés 34

1.3.1 Définitions et notations 34

1.3.2 Classes de graphes 36

1.3.3 Séparateurs minimaux 40

1.3.4 Décomposition par séparateurs minimaux complets 43

1.3.5 Triangulation minimale 45

I PHYLOGENIE **47**

Introduction de la première partie **49**

2 Dissimilarité : interprétation par des graphes **51**

2.1 De la dissimilarité à la famille de graphes 51

2.1.1 Dissimilarités, distances, représentations arborées 52

2.1.2 Matrice ordinale associée à une dissimilarité 54

2.1.3 Famille de graphes emboîtés associée à une matrice ordinale 58

2.2 Les distances triangulées 59

3 Ajustements de données **63**

3.1 Ajuster les données en triangulant 64

3.1.1 Un schéma de composition de graphes par arêtes 64

3.1.2 Un nouveau concept de graphes : la sous-triangulation maximale 67

3.1.3 Un algorithme de sous-triangulation de la famille de graphe définie par une matrice ordinale 68

3.2 Etude combinatoire des configurations compatibles avec une distance additive 71

3.2.1 Configurations des familles de graphes instanciant une matrice ordinale additive 71

3.2.2 Règles extraites de l'étude des configurations à 4 sommets 76

3.3 Perspectives 79

Conclusion de la première partie **81**

II ANALYSE FORMELLE DE CONCEPTS **83**

Introduction de la deuxième partie **85**

4 Codage d'un treillis par un graphe **87**

4.1 Un nouveau codage du treillis des concepts 88

4.1.1 Graphe co-biparti associé à une relation 88

4.1.2 Codage d'un treillis des concepts 92

4.1.3 Sommets universels et sommets simpliciaux 95

4.2 Sélection d'un sous-treillis par saturation de séparateurs minimaux du graphe sous-jacent 99

4.2.1 Effet de la saturation d'un séparateur minimal du graphe sous-jacent 99

4.2.2 Equivalence entre chaînes maximales du treillis et triangulations minimales du graphe 101

4.3 Décomposition par séparateurs minimaux d'une relation et de son treillis . 103

4.4 Comment assurer que le treillis d'une relation soit de taille polynomiale . . 106

5 La domination : une notion clef 113

5.1 La domination classique 113

5.2 Propriétés remarquables de la domination 115

5.2.1 Domination dans un graphe quelconque 115

5.2.2 Domination dans le graphe sous-jacent à une relation 118

5.3 Moplex et atomicité 121

5.3.1 Maxmods et Moplex dans un graphe quelconque 121

5.3.2 Equivalence entre atomicité dans un treillis et moplex du graphe sous-jacent 121

5.3.3 Calcul de la couverture d'un élément 123

5.3.4 Comment un concept hérite de la domination de ses ancêtres 125

5.4 La table de domination 125

5.4.1 Représentation de la domination par une table 126

5.4.2 Requêtes sur la table de domination 127

5.4.3 Mise à jour de la table de domination 127

5.4.4 Une table simplifiée : la table de cardinalité 129

6 Applications algorithmiques 131

6.1 Maintien d'une sous-hiérarchie de Galois 131

6.1.1 Treillis des concepts et sous-hiérarchies de Galois 132

6.1.2 Utilisation de la domination pour décomposer une sous-hiérarchie de Galois 134

6.1.3 Mise à jour des informations de domination 138

6.2 Génération du treillis des concepts 142

6.2.1 Algorithmes existants 143

6.2.2 Génération des concepts par calcul des séparateurs minimaux . . . 144

6.2.3 Génération des concepts par le calcul de la couverture 144

Conclusion de la deuxième partie **157**

Conclusions générales et perspectives **161**

Bibliographie **162**

Index **169**

Introduction : une passerelle entre plusieurs domaines

Il y a deux ans, je me suis donné pour but, au cours de ce travail de thèse, d'étudier les applications actuelles les plus vitales de l'Informatique et d'essayer d'y apporter ma contribution selon mes intérêts et mes compétences. J'ai choisi comme champ d'investigation deux grands domaines qui s'offraient à moi, de par l'environnement de recherche du LIMOS et de l'université Blaise Pascal : la bio-informatique et les bases de données.

Ce travail s'inscrit dans une perspective de Data Mining, domaine qui a pour propos d'analyser de vastes bases de données, dans le but d'établir des relations non triviales et de trouver de nouvelles présentations synthétiques des informations ([HMS01]). Les arbres et les treillis, et plus généralement les graphes, sont des structures intéressantes dans cette optique, car elles mettent en évidence l'organisation intrinsèque des données et sont faciles à visualiser.

Comme Jean-Paul Bordat, dont j'ai été l'élève, je suis avant tout algorithmicien. Je me situe dans une approche méthodologique, que Brassard et Bratley qualifient d'approche *a priori* ([BB87]), qui évalue selon des critères mathématiques rigoureux la quantité de ressources informatiques nécessaires aux algorithmes en fonction de la taille des données à traiter. Par opposition à l'approche empirique (ou *a posteriori*), qui teste les algorithmes sur divers exemplaires à l'aide d'ordinateurs, l'approche *a priori* se concentre sur les difficultés essentielles de la conception algorithmique. Quand ces difficultés se révèlent importantes, cette approche permet, en différant les efforts d'implémentation, d'éviter de se perdre dans des solutions inefficaces.

Cette thèse se veut également une illustration de la puissance des graphes (non orientés et non valués), trop souvent considérés comme des objets abstraits, sans application. Or nous montrons, pour deux domaines très différents, la Phylogénie et l'Analyse Formelle de Concepts, que l'utilisation des graphes apporte un éclairage nouveau et débouche rapidement sur des résultats algorithmiques.

Les outils que nous utilisons sont spécifiques aux graphes non orientés, qui sont la spécialité de ma co-directrice de thèse, Anne Berry. Ses travaux, empreints d'une approche très combinatoire, sans doute héritée d'Alain Guénoche et de Michel Habib, sont centrés sur les séparateurs minimaux dans les graphes non orientés, que ces résultats soient théo-

riques ou algorithmiques. Ils portent, eux aussi, la marque de la méthode novatrice de Jean-Paul Bordat et les preuves s'y font plus souvent à partir d'invariants que par des méthodes plus classiques. Avec Jean-Paul Bordat, Anne Berry a su, par ailleurs, utiliser les treillis, pour extraire de nombreux résultats sur les graphes.

Paradoxalement, mes propres penchants naturels me poussent plus vers les graphes orientés que les graphes non orientés. Une de mes contributions personnelles, dans cette thèse, est la mise en évidence et l'étude d'une structure ordonnée qui apparaît dans les graphes : celle induite par la relation de domination.

C'est la combinaison de ces deux approches, en fait complémentaires, qui a produit les résultats présentés dans ce travail.

Ce mémoire comporte deux parties qui abordent deux domaines en apparence très différents l'un de l'autre, ce qui, à première vue, pourrait donner une impression de disparité. Il y a cependant des liens forts entre les deux.

Tout d'abord, notre approche commune est d'utiliser une modélisation par des graphes. Un graphe non orienté est une relation binaire symétrique, objet mathématique auquel nous cherchons à nous ramener. Dans la première partie, nous partons d'une relation valuée symétrique (une dissimilarité) entre objets et nous la binarisons. Cette stratégie de binarisation est classique et fréquemment utilisée en Bases de Données et en Data Mining. Dans la deuxième partie, au contraire, nous partons d'une relation binaire et nous la symétrisons, en codant cette relation par un graphe doté de propriétés remarquables.

Dans les deux cas, nous travaillons ensuite sur une famille de graphes emboîtés ; ces graphes sont définis par les seuils d'une matrice ordinale pour la phylogénie, ils sont définis par les éléments rencontrés lors du parcours d'une chaîne maximale du treillis pour l'analyse formelle de concepts.

Dans les deux cas aussi, les procédés de triangulation (plongement dans un graphe triangulé) revêtent une grande importance, avec la notion de "sous-triangulation maximale" dans la première partie, et dans la deuxième l'exploration du treillis en utilisant les chaînes maximales qui ne sont autres que des triangulations minimales du graphe dont nous nous servons pour coder le treillis.

Ce mémoire est organisé comme suit :

Nous commençons par un chapitre général qui rappelle les résultats connus qui nous seront utiles et définit les notations qui seront utilisées dans les chapitres suivants.

La première partie s'intéresse aux applications bio-informatiques et, plus particulièrement, aux mesures de dissimilarités destinées à la construction d'arbres d'évolution. Le chapitre 2 décrit la problématique spécifique que nous abordons et met en place des outils algorithmiques et théoriques permettant de résoudre un problème. Le chapitre 3 décrit la solution algorithmique proposée ; il étudie ensuite des prolongements possibles issus d'une étude combinatoire.

La seconde partie s'intéresse au Data Mining et, plus particulièrement, à l'analyse formelle de concepts. Le chapitre 4 met en place les outils théoriques qui seront nécessaires, en décrivant notre codage d'une relation par un graphe non orienté. Le chapitre 5 étudie

plus spécifiquement l'ordre de domination sur un graphe. Le chapitre 6, enfin, illustre la validité de notre approche en proposant plusieurs applications algorithmiques qui améliorent les résultats existants.

Chapitre 1

Définitions et notations

Ce chapitre a pour but de préciser les notations qui seront utilisées tout au long de ce mémoire et de donner des résultats généraux sur lesquels s'appuie ce travail, pour autant qu'ils ne soient pas particuliers à un chapitre. Nous établissons aussi quelques propriétés élémentaires qui ont un intérêt général pour les objets définis et sur lesquelles nous nous appuierons dans les chapitres suivants.

1.1 Relations binaires

Après quelques précisions sur les notations ensemblistes utilisées, nous rappelons des notions fondamentales sur les relations, ainsi que sur les graphes orientés. Nous en profitons pour présenter deux structures de données indispensables pour notre étude : les tables et les matrices.

1.1.1 Notations ensemblistes

Nous rappelons ici, pour les lecteurs peu familiarisés, les conventions ensemblistes généralement adoptées.

Tous les ensembles considérés seront, sauf mention contraire, **finis**, même si certains résultats ou définitions pourraient être étendus au cas infini. Le cardinal d'un ensemble E sera notée $|E|$.

L'opérateur $\subseteq$ sera utilisé pour l'inclusion large tandis que $\subset$ dénotera l'inclusion stricte. Nous utiliserons aussi les opérateurs duaux $\supseteq$ et $\supset$.

X est un sous-ensemble **propre** de E quand $X \subset E$ et $X \neq \emptyset$. Deux ensembles sont dits disjoints si leur intersection est vide.

L'opérateur ensembliste $+$ pourra remplacer le symbole $\cup$ pour l'**union disjointe** : $X+Y = X \cup Y$ avec $X \cap Y = \emptyset$.

L'opérateur ensembliste $-$ dénotera la différence entre deux ensembles : $X-Y = \{x \in X \mid x \notin Y\}$; notons que cette définition n'oblige pas Y à être inclus dans X.

La notation $[p..q]$, inspirée du langage Pascal, correspondra à un intervalle entier : $[p, q] \subseteq \mathbb{Z}$.

Pour deux ensembles X et Y, le **produit cartésien** $\{(x, y) | x \in X,\ y \in Y\}$ sera noté $X \times Y$. Le produit cartésien $X \times X$ sera aussi noté X^2.

L'ensemble des parties d'un ensemble X sera noté 2^X.

Une **partition** d'un ensemble E est une famille $\mathcal{F} \subset 2^X$ de sous-ensembles de E deux à deux disjoints et dont l'union est égale à E. On dit qu'une partition $\mathcal{F}$ sur E est **plus fine** qu'une partition $\mathcal{F}'$ sur E quand $\forall E_i \in \mathcal{F}$, $\exists E'_i \in \mathcal{F}'$, $E_i \subseteq E'_i$.

1.1.2 Relations

La notion de relation fait partie des notions fondamentales des mathématiques. Elle permet d'établir des liens entre les éléments d'un ou de plusieurs ensembles. Par son pouvoir de modélisation, elle est aussi devenue un outil essentiel de l'informatique, en particulier pour les bases de données et le data mining. Nous allons préciser pour les relations binaires le vocabulaire et les notations que nous utiliserons dans ce mémoire.

Définition 1.1.1 RELATION.

- *On appelle* ***relation binaire*** *sur un couple (X, Y) d'ensembles tout sous-ensemble $\mathcal{R}$ du produit cartésien $X \times Y$. On dira que $\mathcal{R}$ est une relation* **sur** (X, Y)
- *Nous appellerons* ***relation partielle*** *d'une relation $\mathcal{R}$ sur (X,Y) toute relation sur (X,Y) incluse au sens de l'inclusion de deux parties d'un produit cartésien dans $\mathcal{R}$.*
- *Nous appellerons* ***sous-relation*** *d'une relation $\mathcal{R}$ sur (X,Y), toute relation $\mathcal{R} \cap (A \times B)$, notée $\mathcal{R}(A, B)$, définie sur $A \subseteq X$ et $B \subseteq Y$.*

Par abus de langage, nous utiliserons le terme sous-relation au lieu de relation partielle quand il n'y a pas d'ambiguïté. Nous noterons $|\mathcal{R}|$ la taille d'une relation $\mathcal{R}$.

Définition 1.1.2 TABLE D'UNE RELATION.
Soit $\mathcal{R}$ une relation sur (X, Y). Le tableau booléen T à deux dimensions, tel que $\forall x \in X, y \in Y$, $T[x, y] = 1 \Longleftrightarrow (x, y) \in \mathcal{R}$, est appelé ***table de la relation*** *$\mathcal{R}$.*

Dans ce mémoire, nous représenterons la table avec les éléments de X sur les colonnes et les éléments de Y sur les lignes. Les cases $T[x, y] = 1$ seront cochées par $\times$ et les cases $T[x, y] = 0$ seront laissées vides.

Nous appellerons **un** ou **croix** d'une relation $\mathcal{R}$ sur (X, Y) tout un couple (x, y) de $X \times Y$ appartenant à $\mathcal{R}$. Une **rangée de uns** sera représentée par un élément x de X

tel que $\forall y \in Y$, $(x, y) \in \mathcal{R}$ (colonne de uns) ou par un élément y de Y tel que $\forall x \in X$, $(x, y) \in \mathcal{R}$ (ligne de uns) ; nous définissons de façon similaire les **zéros** ($(x, y) \notin \mathcal{R}$) et les **rangées de zéros**.

Exemple 1.1.3
Table d'une relation $\mathcal{R}$ sur (X, Y), $X = \{a, b, c, d, e\}$ et $Y = \{x, y, z\}$.

$\mathcal{R}$	a	b	c	d	e
x	×	×			
y	×	×		×	×
z	×			×	

Table de $\mathcal{R} = \{(a, x), (a, y), (a, z), (b, x), (b, y), (d, y), (d, z), (e, y)\}$.

$\mathcal{R}$ possède une colonne de uns : a, et une colonne de zéros : c ; $\mathcal{R}$ ne possède ni ligne de uns ni ligne de zéros.
◇

Définition 1.1.4 DUALITÉS.
*La **réciproque** d'une relation $\mathcal{R}$ sur (X, Y) est la relation, notée $\mathcal{R}^{-1}$, sur (Y, X) telle que $\forall x \in X$, $\forall y \in Y$, $(x, y) \in \mathcal{R}^{-1} \iff (y, x) \in \mathcal{R}$.*
*La **complémentaire** d'une relation $\mathcal{R}$ sur (X, Y) est la relation, notée $\bar{\mathcal{R}}$, sur (X, Y) telle que $\forall x \in X$, $\forall y \in Y$, $(x, y) \in \bar{\mathcal{R}} \iff (x, y) \notin \mathcal{R}$.*

La table de la relation réciproque $\mathcal{R}'$ est obtenue en prenant la transposée – au sens des matrices – de la table de $\mathcal{R}$. La table de la relation complémentaire $\bar{\mathcal{R}}$ est obtenue en remplaçant les zéros par des uns et réciproquement.

Exemple 1.1.5
Réciproque et complémentaire de la relation $\mathcal{R}$ de l'exemple 1.1.3.

$\mathcal{R}^{-1}$	x	y	z
a	×	×	×
b	×	×	
c			
d		×	×
e		×	

$\mathcal{R}^{-1}$ réciproque de $\mathcal{R}$

$\bar{\mathcal{R}}$	a	b	c	d	e
x			×	×	×
y			×		
z		×	×		×

$\bar{\mathcal{R}}$ complémentaire de $\mathcal{R}$

◇

Définition 1.1.6 RELATION INTERNE.
*Une relation binaire $\mathcal{R}$ sur (X, Y) est dite **interne** quand $X = Y$. On dira alors que $\mathcal{R}$ est une relation **sur** X.*

Définition 1.1.7 PROPRIÉTÉS DES RELATIONS INTERNES.
Une relation $\mathcal{R}$ sur X est :
- ***réflexive*** *quand* $\forall x \in X$, $(x, x) \in \mathcal{R}$,
- ***irréflexive*** *quand* $\forall x \in X$, $(x, x) \notin \mathcal{R}$,
- ***symétrique*** *quand* $\forall x, y \in X$, $(x, y) \in \mathcal{R} \Longrightarrow (y, x) \in \mathcal{R}$,

- ***antisymétrique*** *quand* $\forall x, y \in X, x \neq y$, $(x, y) \in \mathcal{R} \Longrightarrow (y, x) \notin \mathcal{R}$,
- ***transitive*** *quand* $\forall x, y, z \in X$, (x, y), $(y, z) \in \mathcal{R} \Longrightarrow (x, z) \in \mathcal{R}$.
- ***acyclique*** *quand elle est irréflexive et que, pour toute suite* $(x_1, ..., x_k)$ *de* $k \geq 1$ *éléments de* X *telle que* $\forall i \in [2..k]$ $(x_{i-1}, x_i) \in \mathcal{R}$, *on a* $(x_k, x_1) \notin \mathcal{R}$.

Théorème 1.1.8
Toute relation acyclique est antisymétrique.

Définition 1.1.9 *Table simplifiée*
La table d'une relation irréflexive et symétrique peut être ***simplifiée*** *en ne conservant qu'une demi-table sans diagonale sur le modèle ci-dessous.*

Relation $\mathcal{R} = \{(a,b),(a,d),(b,d),(c,d),(b,a),(d,a),(d,b),(d,c)\}$ *définie sur* $X = \{a, b, c, d\}$.

	a	b	c	d
a		$\times$		$\times$
b	$\times$			$\times$
c				$\times$
d	$\times$	$\times$		$\times$

Table de $\mathcal{R}$

	b	c	d
a	$\times$		$\times$
b			$\times$
c			$\times$

Table simplifiée de $\mathcal{R}$

Définition 1.1.10 PRÉORDRE, RELATION D'ÉQUIVALENCE.
On appelle **préordre** *une relation interne qui est réflexive et transitive.*
On appelle **relation d'équivalence** *une relation interne qui est réflexive, symétrique et transitive.*

1.1.3 Graphes orientés

L'acte de naissance des graphes peut être situé en 1737 avec la résolution par Euler du problème des *sept ponts de Königsberg* : pouvait-on parcourir la ville en passant exactement une fois sur chaque pont ? Pour apporter une réponse négative à cette question, Euler traça un plan de la ville réduit à son plus élémentaire squelette : des points (des "sommets", représentant les différentes parties de la ville) reliés par des traits (des "arêtes", représentant les ponts) ; il avait compris, au delà de l'apparence géométrique, que le problème était de nature topologique : seul importait la continuité du tracé des arêtes entre les sommets. La théorie des graphes distingue les graphes orientés (dans lesquels les "arêtes" prennent le nom d'arc), que nous allons aborder ici et qui comprennent les structures ordonnées que nous aborderons dans la section suivante, des graphes non orientés qui feront l'objet de la fin de ce chapitre.

Définition 1.1.11 GRAPHE ORIENTÉ.
Un ***graphe orienté*** $G = (V, A)$ *est défini par la donnée d'un ensemble fini* V *de sommets et d'un ensemble* $A \subseteq V^2$ *d'arcs* (x, y) *notés* xy.
Quand $xy \in A$, *on dit que* y *est un* ***successeur*** *de* x *et que* x *est un* ***prédécesseur*** *de* y.

Définition 1.1.12 MATRICE D'INCIDENCE D'UN GRAPHE ORIENTÉ.
Sur le même principe que la table d'une relation interne, on définit la ***matrice d'incidence*** *d'un graphe orienté $G = (V, A)$ comme un tableau booléen M à deux indices tel que $\forall x, y \in V$, $M[x, y] = 1 \Longleftrightarrow xy \in A$.*

Nous allons avoir besoin de la notion de stable, qui sera définie de façon similaire pour les graphes non orientés (voir définition 1.3.3).

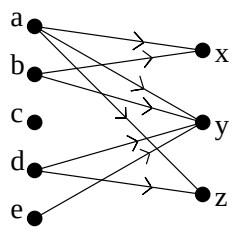
Définition 1.1.13 STABLE.
Un graphe orienté est un **stable** *quand il n'a aucun arc.*

Plus généralement, on peut faire correspondre un graphe orienté à toute relation sur un couple (X, Y) d'ensembles finis :

Définition 1.1.14 GRAPHE BIPARTI *(version orientée).*
Un graphe orienté $G = (V, A)$ est dit ***biparti*** *si son ensemble de sommets V peut être partitionné en deux stables X et Y tels que tous les arcs $xy \in A$ relient un sommet x de X et un sommet y de Y.*

Définition 1.1.15 BIPARTI ORIENTÉ associé à une relation binaire *(voir [Bor86]).*
A une relation $\mathcal{R}$ sur (X, Y), on peut associer un graphe orienté biparti $H = (V, A)$ défini par : $V = X+Y$ et $xy \in A \Longleftrightarrow (x, y) \in \mathcal{R}$.

Exemple 1.1.16
Le graphe biparti orienté associé à la relation $\mathcal{R}$ de l'exemple 1.1.3 :



◇

Dans un graphe orienté $G = (V, A)$, nous appellerons **chaîne** toute suite $(x_1, ..., x_k)$ de sommets distincts tels que $\forall i \in [2..k]$, $x_{i-1}x_i \in A$. On dit alors que x_k est un **descendant** de x_1. Une chaîne $(x_1, ..., x_k)$ forme un **circuit** quand $x_k x_1 \in A$. Une chaîne est dite **maximale** si elle n'est incluse proprement dans aucune autre chaîne.

Définition 1.1.17 ARBORESCENCE.
Dans un graphe orienté, nous dirons qu'un sommet r est une ***racine*** *quand, pour tout sommet x du graphe, il existe une* ***unique*** *chaîne $(r, ..., x)$.*
Une arborescence est un graphe orienté qui possède une unique racine.

Rappelons enfin, pour le lecteur peu familiarisé avec l'algorithmique de graphes, le procédé fondamental de parcours d'un graphe orienté. L'algorithme `DESCENDANCE` calcule

l'ensemble des descendants d'un sommet x_0 d'un graphe G. L'algorithme `PARCOURS` permet d'atteindre tous les sommets du graphe.

Algorithme 1.1.18 `DESCENDANCE`

Donnée : Un graphe orienté $G = (V, A)$, un sommet $x_0 \in V$.
Résultat : ATTEINT est l'ensemble des descendants de x_0 (x_0 compris).
Initialisation
ATTEINT $\leftarrow \{x_0\}$;
EXPLORÉ $\leftarrow \emptyset$;
Début
Tant que ATTEINT−EXPLORÉ$\neq \emptyset$ **faire :**
  Choisir y dans ATTEINT−EXPLORÉ ;
  Si tous les successeurs de y sont dans ATTEINT
    alors EXPLORÉ $\leftarrow$ EXPLORÉ$\cup\{y\}$;
    sinon :
      Choisir un successeur $z \notin$ ATTEINT de y ;
      ATTEINT $\leftarrow$ ATTEINT$\cup\{z\}$;
Fin.

Algorithme 1.1.19 `PARCOURS`

Donnée : Un graphe orienté $G = (V, A)$.
Résultat : ATTEINT$= V$.
Initialisation
ATTEINT $\leftarrow \emptyset$;
EXPLORÉ $\leftarrow \emptyset$;
Début
Tant que ATTEINT$\neq V$ **faire :**
  Choisir x_0 dans $V-$ATTEINT ;
  ATTEINT $\leftarrow$ ATTEINT$\cup\{x_0\}$;
  // *Appel à* `DESCENDANCE` *sans Initialisation de* ATTEINT *ni* EXPLORÉ.
  `DESCENDANCE`(G, x_0) ;
Fin.

L'ensemble ATTEINT−EXPLORÉ peut être implémenté par un file ou une pile. Dans le premier cas, le parcours se fait **en largeur** : pour explorer les descendants d'un sommet, on commence par atteindre tous ses successeurs et on poursuit de manière concentrique. Dans le second cas, le parcours se fait **en profondeur** : on explore toute la descendance d'un premier successeur d'un sommet avant de passer à un autre successeur.

Le parcours d'un graphe peut se faire en temps linéaire $O(n + m)$.

De nombreux algorithmes de graphes sont basés sur un schéma de parcours.

1.2 Ensembles ordonnés

Nous rappelons ici des notions fondamentales sur les ensembles ordonnés en général et plus particulièrement sur les treillis. Pour ces derniers, nous nous attardons sur une

structure fondamentale pour notre travail : les treillis de Galois, après avoir mentionné très brièvement la notion de fermeture qui lui donne un fondement mathématique. Nous utiliserons pour les structures ordonnées une partie du vocabulaire et des notations définies précédemment pour les graphes orientés.

1.2.1 Ordres

Définition 1.2.1 ORDRES.
• *Un* ***ordre large*** *est un couple* $(P,\leq)$*, où* P *est un ensemble et* $\leq$ *une relation sur* P *qui est* ***réflexive****, antisymétrique et transitive.*
• *Un* ***ordre strict*** *est un couple* $(P,<)$*, où* P *est un ensemble et* $<$ *une relation sur* P *qui est* ***irréflexive****, antisymétrique et transitive.*

Dans les deux cas, on dit que P est un ensemble ordonné (*partial ordered set*, en abrégé *poset*).

Pour tous les éléments x et y de P tels que $x\leq y$ (ou $x<y$), nous dirons que x est un **ascendant** de y et que y est un **descendant** de x.

Théorème 1.2.2 ORDRE QUOTIENT CANONIQUE.
Soit $\mathcal{R}$ *un préordre sur un ensemble* X*. On définit une relation d'équivalence* $\equiv$ *entre sommets par* $x \equiv y \iff (x,y) \in \mathcal{R} \wedge (y,x) \in \mathcal{R}$*. La partition de* X *en classes d'équivalence de* $\equiv$ *définit alors une relation d'ordre qu'on appellera* ***ordre quotient canonique*** *du préordre* $\mathcal{R}$*.*

Exemple 1.2.3
Ordre canonique associé à un préordre.

$\mathcal{R}$	a	b	c	d	e
a	×				
b	×	×	×		
c	×	×	×		
d	×	×	×	×	
e	×				×

Un préordre $\mathcal{R}$
défini sur $X = \{a,b,c,d,e\}$

$\mathcal{R}'$	a	**bc**	d	e
a	×			
bc	×	×		
d	×	×	×	
e	×			×

L'ordre canonique associé $\mathcal{R}'$,
défini sur $X' = \{a,bc,d,e\}$

◇

Définition 1.2.4 COUVERTURE.
Soient $(P,\leq)$ *un ordre,* x *et* y *deux éléments distincts de* P*. On dit que* x ***est couvert par*** y *ou encore que* y ***couvre*** x *quand* $x\leq y$ *et il n'existe aucun autre élément* z *de* P *tel que* $x\leq z$ *et* $z\leq y$*. La relation ainsi définie sur* P *est appelée* ***couverture*** *de l'ordre.*
La ***couverture de*** $x\in P$ *est l'ensemble des* y *de* P *tels que* x *est couvert par* y*.*
Quand y *couvre* x*, nous dirons que* x ***est un prédécesseur*** *de* y *et que* y ***est un successeur de*** x*.*

La relation de couverture est donc obtenue en supprimant toutes les boucles de réflexivité et tous les arcs de transitivité dans un ordre. Ce procédé permet d'alléger les représentations tout en conservant toutes les informations pertinentes de la structure. Le graphe orienté ainsi obtenu porte le nom de diagramme de Hasse (voir [Bor92]) ou plus simplement de diagramme ([Bir67]). Dans l'usage courant, on confond souvent un ordre avec sa couverture.

Sauf précision contraire, dans toute la suite, nous choisirons de représenter un diagramme de Hasse en plaçant les plus grands éléments en haut et les plus petits en bas (les successeurs au dessus des prédécesseurs, $\top$ au sommet et $\perp$ à la base) ; le sens des arcs est ainsi implicite. Les conventions dans ce domaine varient suivant les auteurs.

Le procédé réciproque du passage à la couverture, applicable à une relation acyclique, porte le nom de **fermeture réflexo-transitive** (c'est une fermeture au sens de la définition 1.2.36). Le schéma inductif correspondant a pour base une relation acyclique $\mathcal{R}$ sur un ensemble X et pour règles :
1) Pour $x \in X$, ajouter (x, x) à $\mathcal{R}$;
2) Pour $x, y, z \in X$ tels que $(x, y), (y, z) \in \mathcal{R}$, ajouter (x, z) à $\mathcal{R}$.

Exemple 1.2.5
Relation de couverture de l'ordre $\mathcal{R}'$ de l'exemple 1.2.3.

	a	**bc**	d	e
a				
bc	×			
d		×		
e	×			

Table de la couverture de $\mathcal{R}'$

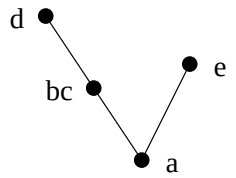


Diagramme de Hasse de $\mathcal{R}'$

En prenant la fermeture réflexo-transitive de cette couverture, on retrouve l'ordre $\mathcal{R}'$.
◇

Définition 1.2.6 SOUS-ORDRES.
*On appelle **sous-ordre** d'un ordre $(P, \leq)$, toute partie $Q \subseteq P$ qui est aussi un ordre.*

Théorème 1.2.7 PRINCIPE DE DUALITÉ *(ordres).*
Soit $(P, \leq)$ un ordre. La relation réciproque $\geq$ sur P définie par $x \geq y \Longleftrightarrow y \leq x$ est une relation d'ordre sur P. $(P, \geq)$ est appelé ordre **dual** *de $(P, \leq)$.*

Définition 1.2.8 ORDRE IPSODUAL.
*Un ordre est dit **ipsodual** quand il est isomorphe à son dual.*

Définition 1.2.9 COMPARABILITÉ.
*Deux éléments x et y d'un ordre $(P, \leq)$ sont dit **comparables** quand $x \leq y$ ou $y \leq x$; ils sont dits **incomparables** sinon. Une partie non vide de P dont tous les éléments sont deux à deux incomparables est appelée **antichaîne** de P.*

Une antichaîne d'un ordre est dite **maximale** si elle n'est proprement incluse dans aucune autre antichaîne de l'ordre.

Définition 1.2.10 ORDRE TOTAL.
*Un ordre $(P, \leq)$ est dit **total** quand tous les éléments de P sont comparables deux à deux. On dit alors que P est une **chaîne**.*
Plus généralement, on appelle chaîne dans un ordre $(P, \leq)$ quelconque, toute partie de P formant un sous-ordre total.

Une chaîne d'un ordre P peut donc être vue comme une suite $(x_1, ..., x_k)$ d'éléments de P tels que $\forall i \in [2..k]$ x_i couvre par x_{i-1} ; ceci nous ramène à la définition d'une chaîne dans un graphe orienté quelconque.

Une chaîne d'un ordre est dite **maximale** si elle n'est proprement incluse dans aucune autre chaîne de l'ordre.

Exemple 1.2.11
Le diagramme ci-dessous représente un ordre ipsodual $(P, \leq)$,
avec $P = \{a, b, c, d, e, f, g, h, i, j, k, l, m\}$.

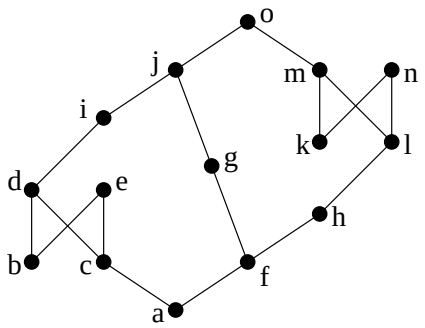


Diagramme de Hasse de P

La permutation $\{a \leftrightarrow o, b \leftrightarrow n, c \leftrightarrow m, d \leftrightarrow l, e \leftrightarrow k, f \leftrightarrow j, g \leftrightarrow g\}$ est un isomorphisme qui associe $(P, \leq)$ à son dual $(P, \geq)$.
f et m sont comparables $(f \leq m)$; e et m sont incomparables.
$afhlmo$ et ace sont des chaînes maximales ; $defk$ et $bcghk$ sont des antichaînes maximales.
$\diamond$

Définition 1.2.12 ORDRE GRADUÉ.
*Un ordre $(P, \leq)$ est dit **gradué** quand il existe une application g de P vers $\mathbb{N}$ telle que :*

1. *g est monotone croissante : si $x \leq y$ alors $g(x) \leq g(y)$, et*

2. *pour tous sommets x et y de P, si y couvre x alors $g(y) = g(x) + 1$.*

Exemple 1.2.13
Le diagramme ci-dessous représente un ordre gradué $(Q, \leq)$
avec $Q = \{a, b, c, d, e, f, g, h, i, j, k\}$.

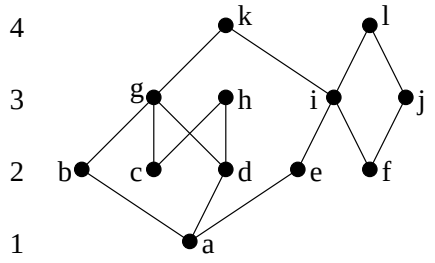


Diagramme de Hasse de Q

Sur la figure ci-dessus, les éléments de l'ordre sont dessinés en niveaux successifs du niveau 1 pour a au niveau 4 pour k et l.
◇

Définition 1.2.14 ELEMENTS MAXIMAUX, MINIMAUX.
• *Un élément x d'un ordre $(P, \leq)$ est dit* ***maximal*** *si P n'admet aucun élément plus grand que x ; c'est-à-dire $\forall y \in P - \{x\}$, $x \not\leq y$.*
• *Un élément x d'un ordre $(P, \leq)$ est dit* ***minimal*** *si P n'admet aucun élément plus petit que x ; c'est-à-dire $\forall y \in P - \{x\}$, $y \not\leq x$.*

Définition 1.2.15 MAXIMUM, MINIMUM.
• *Un ordre $(P, \leq)$ admet x pour* ***maximum*** *si x est plus grand que tous les autres éléments de P ; c'est-à-dire $\forall y \in P - \{x\}$, $y \leq x$.*
• *Un ordre $(P, \leq)$ admet x pour* ***minimum*** *si x est plus petit que tous les autres éléments de P ; c'est-à-dire $\forall y \in P - \{x\}$, $x \leq y$.*

Remarque 1.2.16
Le maximum (resp. minimum) – ***s'il existe*** *– d'un ordre est unique ; il est conventionnellement appelé* ***Top*** *et noté $\top$ (resp.* ***Bottom*** *et $\perp$).*
Le maximum et le minimum, s'ils existent, appartiennent à toutes les chaînes maximales de l'ordre.
Cette propriété n'est bien évidemment utilisable que si l'ordre possède un maximum. Ce n'est pas toujours le cas, comme le montre l'ordre de l'exemple 1.2.11 qui possède plusieurs éléments maximaux : e, n et o ; la chaîne maximale $afhlmo$ ne contient ni e ni n.

Définition 1.2.17 MAJORANT, MINORANT.
Un élément x d'un ordre $(P, \leq)$ admet y pour ***majorant*** *(resp.* ***minorant****) quand $x \leq y$ (resp. $y \leq x$). Nous noterons $Maj(x)$ l'ensemble des majorants de x et $Min(x)$ l'ensemble de ses minorants. Pour toute partie non vide A de P, on définit $Maj(A) = \bigcap_{x \in A} Maj(x)$ et $Min(A) = \bigcap_{x \in A} Min(x)$.*

On peut écrire $\mathrm{Maj}(A) = \{y \in P \mid \forall x \in A \; x \leq y\}$. Si P a un minimum et un maximum, on a $\mathrm{Maj}(P) = \{\top\}$ et $\mathrm{Min}(P) = \{\perp\}$.

Définition 1.2.18 BORNES SUPÉRIEURE, INFÉRIEURE ; BOTTOM, TOP.
• *On appelle* ***borne supérieure*** *(on trouve aussi les termes : supremum, union,* join, lowest upper bound*) d'une partie A non vide d'un ordre P le minimum, s'il existe de $Maj(A)$ (c'est-à-dire le plus petit majorant). Notations : $Sup(A)$, $Join(A)$, $\bigvee A$, voire*

$x \vee y$ si $A = \{x, y\}$.
• *On définit symétriquement la* ***borne inférieure*** *(on trouve aussi les termes : infimum, inter,* meet, greatest lower bound*) de A comme le plus grand minorant. Notations : Inf(A), Meet(A), $\bigwedge A$, voire $x \wedge y$ si $A = \{x, y\}$.*
• *Dans ce mémoire, nous utiliserons les notations Inf(A) et Sup(A).*

Si P a un maximum $\top$ et un minimum $\bot$, on a donc $\mathrm{Inf}(P) = \bot$ et $\mathrm{Sup}(P) = \top$.

Exemple 1.2.19
Avec les ordres P de l'exemple 1.2.11 et Q de l'exemple 1.2.13 :
• P a pour éléments maximaux e, o et n et pour éléments minimaux a, b et k.
$Maj(f) = \{f, g, h, j, l, m, n, o\}$, $Maj(d) = \{d, i, j, o\}$; $Maj(d, f) = \{j, o\}$ a pour minimum j qui est donc la borne supérieure de $\{d, f\}$. $Maj(\{b, c\}) = \{d, e, i, j, o\} = Maj(c)$, cet ensemble n'a pas de minimum (mais deux éléments minimaux d et e) donc $\{b, c\}$ n'a pas de borne supérieure.
P n'a ni maximum ni minimum.
• Q a pour éléments maximaux h, k et l et pour éléments minimaux a, c et f.
$Maj(b, e) = \{b, g, k\} \cap \{e, i, k, l\} = \{k\}$, $\{b, e\}$ a donc pour borne supérieure k. $Maj(b, j) = \{b, g, k\} \cap \{j, l\} = \emptyset$, $\{b, j\}$ n'a donc pas de borne supérieure.
Q n'a ni maximum ni minimum.
◇

1.2.2 Treillis

Les notions de bornes supérieure et inférieure généralisent la notion de plus proche ancêtre commun (*nearest common ancestor*) dans une arborescence. Les treillis peuvent ainsi être vus comme une extension des arborescences, la notion de plus proche ancêtre commun y étant en quelque sorte dédoublée en borne inférieure et borne supérieure. Le lecteur trouvera une étude détaillée des treillis dans les ouvrages classiques [Bir67] et [BM70].

Définition 1.2.20 TREILLIS.
Un ordre $(P, \leq)$ est un demi-treillis supérieur si toute paire d'éléments de P possède une borne supérieure. Un ordre $(P, \leq)$ est un demi-treillis inférieur si toute paire d'éléments de P possède une borne inférieure.
On appelle ***treillis*** *un ordre qui est à la fois demi-treillis supérieur et demi-treillis inférieur.*

Définition 1.2.21 ATOMES ET CO-ATOMES.
On appelle ***atome*** *dans un treillis tout élément couvrant bottom. On appelle* ***co-atome*** *dans un treillis tout élément couvert par top.*

Définition 1.2.22 IRRÉDUCTIBLES.
Un élément (différent de $\top$) d'un treillis est dit ***sup-irréductible*** *quand il couvre exactement un élément. Un élément (différent de $\bot$) d'un treillis est dit* ***inf-irréductible***

quand il est couvert par exactement un élément. Un élément est dit ***irréductible*** *s'il est sup-irréductible ou inf-irréductible.*

Autrement dit, un sup-irréductible x d'un treillis L n'est la borne supérieure d'aucune partie non vide de $L-\{x\}$. De même, un inf-irréductible y n'est la borne inférieure d'aucune partie non vide de $L-\{y\}$.

Définition 1.2.23 TREILLIS COMPLET.
Un treillis $(L, \leq)$ est dit complet quand toute partie non vide de L possède une borne inférieure et une borne supérieure.

Théorème 1.2.24
Tout treillis fini est complet et possède donc un maximum $\top$ et un minimum $\perp$.

Remarque 1.2.25 *Un singleton $\{x\}$ a pour borne supérieure et pour borne inférieure x. Certains auteurs ([GW99]) étendent la notion de bornes supérieure et inférieure à l'ensemble vide : $Inf(\emptyset) = \top$ et $Sup(\emptyset) = \perp$.*

Exemple 1.2.26
Le diagramme ci-dessous représente un treillis $(L, \leq)$
avec $L = \{a, b, c, d, e, f, g, h, i, j, k, l, m\}$.

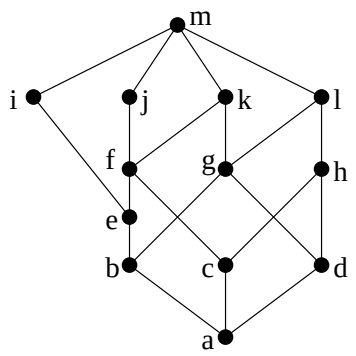


Diagramme de Hasse de L

$\perp = a$, $\top = m$; les atomes sont b, c et d ; les co-atomes sont i, j, k et l.
L'ensemble des sup-irréductibles est $\{b, c, d, e, i, j\}$; l'ensemble des inf-irréductibles est $\{h, i, j, k, l\}$. ni f ni g n'est irréductible.
$abefkm$ et $achlm$ sont des chaînes maximales. $ijkl$ et $fghi$ sont des antichaînes maximales.
$\diamond$

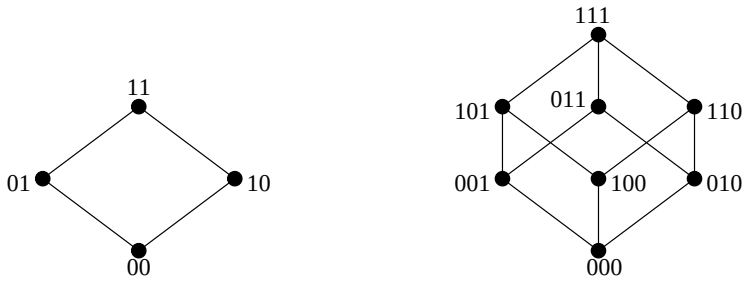
Définition 1.2.27 TREILLIS DISTRIBUTIF.
Un treillis $(L, \leq)$ est dit distributif quand $\forall x, y, z \in L$ $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$.

Définition 1.2.28 TREILLIS COMPLÉMENTÉ.
Dans un treillis, un élément x est appelé un ***complément*** *d'un élément y quand $Sup(\{x, y\}) = \top$ et $Inf(\{x, y\}) = \perp$.*
Un treillis est dit ***complémenté*** *quand chacun de ses éléments a un complément.*

Définition 1.2.29 TREILLIS BOOLÉEN.
Un treillis est dit booléen quand il est distributif et complémenté.

Propriété 1.2.30
Un treillis L est booléen si et seulement si il est isomorphe à un treillis $\{0,1\}^k$ ordonné par : $\forall (a_1,...,a_k),(b_1,...,b_k) \in \{0,1\}^k$, $(a_1,...,a_k) \leq (b_1,...,b_k) \Longleftrightarrow \forall i \in [1..k] a_i \leq b_i$.
Un treillis booléen a donc 2^k éléments, $k \in \mathbb{N}$.
Tout treillis booléen à 2^k éléments est isomorphe au treillis des parties d'un ensemble X à k éléments, ordonnées par inclusion. La notation $(a_1,...,a_k) \in \{0,1\}$ est celle du vecteur caractéristique d'une partie A de $X = \{x_1,...,x_k\}$: $\forall i \in [1..k]$, $x_i \in A \Longleftrightarrow a_i = 1$.

Exemple 1.2.31
Treillis booléens B_4 à $2^2 = 4$ éléments, B_8 à $2^3 = 8$ éléments.



B_4 B_8

◇

Théorème 1.2.32 PRINCIPE DE DUALITÉ *(treillis).*
L'ordre dual d'un treillis est un treillis.

Les notions suivantes sont ainsi duales : bottom et top, atomes et co-atomes, bornes supérieures et bornes inférieures, inf-irréductibles et sup-irréductibles.

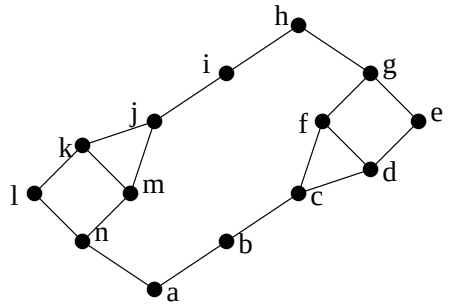
Dans un treillis complémenté, le complément de chaque élément peut être unique ; on peut alors définir une application qui associe à chaque élément son unique complément. Ceci nous amène à la notion d'orthocomplémentation.

Définition 1.2.33 ORTHOCOMPLÉMENTATION *(voir [BB99]).*
*Un Treillis L est dit **orthocomplémenté** s'il existe une application φ de L dans lui-même telle que :*

1. ***Complémentation** : $\forall a \in L$, $Sup(\{a,\varphi(a)\}) = \top$ et $Inf\{a,\varphi(a)\} = \bot$;*
2. ***Dualité** : $\forall x,y \in L$, $\varphi(Sup(\{x,y\})) = Inf(\{\varphi(x),\varphi(y)\})$ et $\varphi(Inf(\{x,y\})) = Sup(\{\varphi(x),\varphi(y)\})$;*
3. ***Involution** : $\forall x \in L$, $\varphi(\varphi(x)) = x$.*

*L'application φ est appelée **orthocomplémentation** de L. Pour tout élément x de L, $\varphi(x)$ est appelé **orthocomplément** de x ou plus simplement **complément**.*

Exemple 1.2.34
Le diagramme ci-dessous représente un treillis L orthocomplémenté.



Le treilis L

La permutation $\{a \leftrightarrow h, b \leftrightarrow i, c \leftrightarrow j, d \leftrightarrow k, e \leftrightarrow l, f \leftrightarrow m, g \leftrightarrow n\}$ est une orthocomplémentation de L.
$\diamond$

1.2.3 Fermetures

Nous allons voir dans la sous-section suivante la notion de treillis de Galois, dont la construction est basée sur un procédé très général de fermeture que nous rappelons rapidement ici et que nous mettons en parallèle avec la notion algorithmique de schéma inductif.

Définition 1.2.35 SYSTÈME DE FERMETURE.
Une famille $\mathcal{F}$ de parties d'un ensemble E est appelé ***système de fermeture*** *quand les deux conditions suivantes sont réunies :*

1. *$\mathcal{F}$ contient E ;*
2. *L'intersection de deux parties de E appartenant à $\mathcal{F}$ appartient aussi à $\mathcal{F}$.*

Définition 1.2.36 OPÉRATEUR DE FERMETURE *([BM70]).*
Une application $\varphi : 2^E \to 2^E$ est appelé ***opérateur de fermeture*** *quand elle vérifie les trois propriétés suivantes :*

1. *Monotonie : $\forall X, Y \in 2^E$, $X \subseteq Y \Longrightarrow \varphi(X) \subseteq \varphi(Y)$;*
2. *Extensivité : $\forall X \in 2^E$, $X \subseteq \varphi(X)$;*
3. *Idempotence : $\forall X \in 2^E$, $\varphi(\varphi(X)) = \varphi(X)$.*

Une fermeture peut être vue comme une formalisation mathématique d'un procédé algorithmique général appelé schéma inductif :

Définition 1.2.37 SCHÉMA INDUCTIF [1].
On dit qu'un ensemble X est défini par un schéma inductif quand on peut le construire en trois phases :

1. *Initialisation : une "base" $B \subseteq X$ donne le contenu minimal de l'ensemble X ;*

[1] D'après J–P. Bordat, cours de graphes et d'algorithmique (licence-maîtrise d'informatique, 1992-1994)

2. *Construction : une famille $\mathscr{F}$ de règles de production définissent les opérations permettant d'ajouter des éléments à l'ensemble en cours de construction ;*
3. *Fermeture : tous les éléments de X peuvent être atteint par une suite d'opérations et seulement ceux-là (X est le plus petit ensemble stable pour ce schéma $(B, \mathscr{F})$).*

1.2.4 Treillis de Galois, treillis des concepts

Les treillis de Galois sont issus de réflexions assez anciennes. Sans qu'il soit utile de remonter jusqu'à Aristote, on peut mentionner la "*Logique de Port Royal*" d'Antoine Arnaud et Pierre Nicole, paru en 1662 sous le titre "La logique ou l'art de penser", et qui se rattache au rayonnement intellectuel de l'abbaye de Port Royal au XVIIième siècle. On trouve, dans ce livre, la première mention explicite de la notion de concept, conçu comme un ensemble d'objets (une *extension*) possédant un certain nombre de propriétés (une *intension*). Les mathématiques s'approprièrent cette notion en définissant, entre deux ensembles en relation, une *correspondance* dite *de Galois*[2] (*Galois connection*, voir [Bir67] et [Ore44]).

Dans le cadre mathématique, une correspondance de Galois peut représenter la structuration en treillis d'un ensemble d'objets en relation avec un ensemble de propriétés, ce qui a été formalisé avec la notion de *treillis de Galois* (voir [BM70]) et redécouvert plusieurs fois. Dans les années 1980, Wille et Ganter ont popularisé les treillis de Galois sous le nom de *treillis des concepts* dans le cadre de ce qu'ils ont appelé "l'*Analyse Formelle de Concepts*" (*Formal Concept Analysis*, en abrégé FCA).

Wille a notamment défini les notions de *contexte formel* et de *concept formel*. Le vocabulaire correspondant étant devenu d'usage courant dans différents domaines de l'informatique, nous l'avons adopté dans ce mémoire. Le lecteur trouvera des précisions sur les bases de la FCA dans ([GW99]).

Dans cette sous-section, nous commençons par rappeler le cadre général des treillis de Galois. Nous abordons ensuite un type particulier de treillis de Galois, le treillis de séparabilité qui a été défini par Berry en liaison avec la séparabilité dans un graphe. Nous terminons avec le nouveau vocabulaire de l'Analyse Formelle de Concepts.

Définition 1.2.38 RECTANGLE MAXIMAL.
Soit $\mathcal{R}$ une relation sur (X, Y). On appelle ***rectangle maximal*** *(ou* ***pavé****) de $\mathcal{R}$ toute partie $(A \times B) \subseteq \mathcal{R}$ telle que : $\forall x \in X - A$, $\exists y \in B \,|\, (x, y) \notin \mathcal{R}$ et $\forall y \in Y - B$, $\exists x \in A \,|\, (x, y) \notin \mathcal{R}$. A est appelé* ***intension*** *(*intent*) du rectangle $A \times B$ et B* ***extension*** *(*extent*) de ce rectangle.*

Ces rectangles maximaux sont parfois appelés fermés (*closed sets*) parce qu'ils forment un système de fermeture (voir définition 1.2.35). L'opérateur de fermeture associé est appelé **fermeture de Galois** de la relation considérée et peut être défini comme une application $\varphi : 2^X \to 2^X$, $\varphi = h \circ g$, avec :

$g : 2^X \to 2^Y$, $g(A) = \{y \in Y \,|\, \forall x \in A, (x, y) \in \mathcal{R}\}$, et

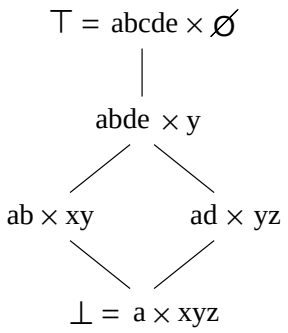
[2]En hommage au mathématicien E. Galois, et par analogie avec sa preuve de l'absence de solution analytique générale des polynômes de degré supérieur à 4.

$h : 2^Y \to 2^X$, $h(B) = \{x \in X \mid \forall y \in B,\ (x, y) \in \mathcal{R}\}$.

Le couple (g, h) est aussi appelé **correspondance de Galois** (*Galois connection*).

Définition 1.2.39 TREILLIS DE GALOIS.
Soit $\mathcal{R}$ une relation sur (X, Y). Le treillis obtenu en ordonnant les rectangles maximaux de $\mathcal{R}$ par inclusion sur les ***intensions*** *est appelé* ***treillis de Galois*** *de $\mathcal{R}$.*

Exemple 1.2.40
Treillis de Galois $\mathcal{L}(\mathcal{R})$ de la relation $\mathcal{R}$ de l'exemple 1.1.3.
a appartient à tous les rectangles maximaux de $\mathcal{R}$; c n'appartient à aucun rectangle maximal, à l'exception de top.
Le calcul du pavé $ab \times xy$ avec la correspondance de Galois se fait comme suit : $g(\{b\}) = \{x, y\}$ donc $h(g(\{b\})) = h(\{x, y\}) = \{a, b\}$.



Le treillis de Galois $\mathcal{L}(\mathcal{R})$

◇

Théorème 1.2.41 PRINCIPE DE DUALITÉ (TREILLIS DE GALOIS) *([Bir67]).*
Le dual d'un treillis de Galois $\mathcal{L}(\mathcal{R})$ associé à une relation $\mathcal{R}$ sur (X, Y) est un treillis de Galois construit sur la relation réciproque $\mathcal{R}^{-1}$ sur (Y, X). Ce treillis dual est isomorphe au treillis obtenu à partir de $\mathcal{R}$ par inclusion sur les extensions de $\mathcal{L}(\mathcal{R})$.

Propriété 1.2.42 *([BM70]).*
Tout treillis est treillis de Galois d'une relation.

Propriété 1.2.43 RELATION RÉDUITE *([BM70]).*
Parmi toutes les relations qui ont le même treillis de Galois (à un isomorphisme près), il en existe une, unique, qui est minimale pour la somme du nombre de lignes et du nombre de colonnes dans la matrice de la relation. Cette relation est dite relation irréductible ou encore ***relation réduite****.*

Caractérisation 1.2.44 *([BM70])* [3].
Une relation est réduite si et seulement si dans sa table :

1. *Il n'y a aucune rangée de uns ;*
2. *Il n'y a pas deux lignes, ni respectivement deux colonnes, identiques.*
3. *Aucune ligne n'est intersection de plusieurs autres lignes, et aucune colonne n'est intersection de plusieurs autres colonnes ;*

On déduit trivialement de ce qui précède une méthode générale pour **réduire** une relation $\mathcal{R}$ sur (X, Y) :

[3] Barbut et Monjardet ont présenté une définition très formelle, nous proposons ici une définition plus sémantique.

Procédé algorithmique 1.2.45
Donnée : La table d'une relation $\mathcal{R} \subseteq (\mathscr{P} \times \mathscr{O})$.
Résultat : La table de la relation réduite correspondante.
Début
Supprimer de la table toutes les rangées de uns ;
Fusionner dans la table les lignes identiques ;
Fusionner dans la table les colonnes identiques ;
Supprimer de la table toute ligne qui est l'intersection de plusieurs autres lignes ;
Supprimer de la table toute colonne qui est l'intersection de plusieurs autres colonnes ;
Fin.

Nous verrons qu'il peut être important de vérifier qu'une relation ne contient ni rangée de uns ni rangée de zéros. En particulier, si une relation $\mathcal{R}$ sur (X, Y) n'a aucune rangée de uns alors $\perp = \emptyset \times Y$ et $\top = X \times \emptyset$.

Barbut et Monjardet ont proposé une démonstration constructive de la propriété 1.2.42, basée sur un schéma algorithmique produisant une relation dont un treillis donné est treillis de Galois.

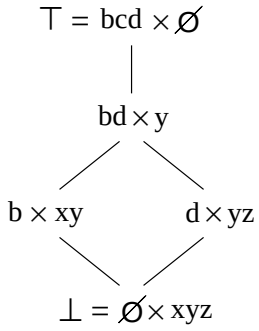
Algorithme 1.2.46 *([BM70])*
Donnée : *Un treillis T.*
Résultat : *Une relation $\mathcal{R} \subseteq X \times Y$ dont T est le treillis de Galois.*
L'étiquetage des éléments de T comme treillis de Galois de $\mathcal{R}$.
Début
// Déterminer les ensembles X et Y sur lesquels construire la relation.
$X \leftarrow$ *ensemble des sup-irréductibles de T ;*
$Y \leftarrow$ *ensemble des inf-irréductibles de T ;*
// Construire la relation $\mathcal{R} \subseteq X \times Y$.
$\mathcal{R} \leftarrow \emptyset$ *;*
Pour chaque *sup-irréductible x* ***faire :***
  Pour chaque *inf-irréductible y* ***faire :***
    si $x \leq y$ **alors** *ajouter (x, y) à $\mathcal{R}$;*
// Etiquetage $A \times B$ des éléments du treillis.
Pour chaque *élément t de T* ***faire :***
  $A_t \leftarrow \{x \in X \mid x \leq t\}$ *;*
  $B_t \leftarrow \{y \in Y \mid y \geq t\}$ *;*
Fin.

Si l'on connaît les degrés entrant et sortant des éléments du treillis, un même parcours en largeur de son diagramme de Hasse permet de construire la relation $\mathcal{R}$ et l'étiquetage des éléments en treillis de Galois. La complexité en temps est linéaire sur la taille du diagramme. La relation construite est une relation réduite.

Exemple 1.2.47
La relation $\mathcal{R}$ de l'exemple 1.1.3 n'est pas réduite : a est une colonne de uns, la colonne e est l'intersection des colonnes b et d. La relation réduite $\mathcal{R}_0$ a un treillis de Galois isomorphe à celui de l'exemple 1.2.40.

	b	c	d
x	×		
y	×		×
z			×

Table de la relation réduite $\mathcal{R}_0$



Treillis de Galois de $\mathcal{R}_0$

Remarquons que la colonne de zéros c a été conservée ; en effet, elle n'était pas intersection d'autres colonnes de $\mathcal{R}$. La suppression de la colonne c de zéros dans $\mathcal{R}_0$ provoquerait la disparition de $bcd \times \emptyset$ dans son treillis de Galois.
◇

La propriété 1.2.42 a permis à Berry et Bordat de caractériser les treillis orthocomplémentés :

Caractérisation 1.2.48 *([BB99]).*
Un treillis fini L possède une fonction d'orthocomplémentation si et seulement si la relation réduite correspondante (relation dont le treillis de Galois est isomorphe à L) est symétrique et irréflexive.

Ces considérations ont permis de définir un type particulier de treillis de Galois, associé à un graphe :

Définition 1.2.49 TREILLIS DE SÉPARABILITÉ *([Ber95], [BB99]).*
*On appelle **treillis de séparabilité** d'un graphe non orienté $G = (V, E)$ le treillis de Galois de la relation complémentaire $V^2 - E$, symétrique et irréflexive, sur V.*

Pour construire un treillis de séparabilité, on considère implicitement que le graphe possède toutes les boucles. Le treillis est alors orthocomplémenté (caractérisation 1.2.48). On peut étendre la définition 1.2.49 à un graphe possédant seulement un certain nombre de boucles (et donc à une relation qui n'est pas irréflexive), comme nous le ferons dans le chapitre 3.

Nous allons terminer cette présentation des treillis de Galois par le vocabulaire de l'Analyse Formelle de Concepts. On s'intéresse toujours à des relations sur une paire d'ensembles. Selon le domaine d'application considéré, les éléments du premier ensemble peuvent être des *propriétés* (Langages à Objets), des *attributs* (Bases de Données), des *questions* (Sciences humaines). Les éléments du second ensemble peuvent être des *objets*, des *tuples* (n-uplets), ou des *individus*.

Définition 1.2.50 TREILLIS DES CONCEPTS *([GW99]).*
*Soient $\mathscr{P}$ un ensemble de "propriétés" et $\mathscr{O}$ un ensemble "d'objets". soit $\mathcal{R}$ une relation sur $(\mathscr{P}, \mathscr{O})$. Le treillis de Galois associé à $\mathcal{R}$ est appelé **treillis des concepts** de $\mathcal{R}$*

et noté $\mathcal{L}(\mathcal{R})$. Le triplet $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ est appelé un ***contexte formel*** *ou plus simplement* ***contexte****. Les rectangles maximaux de $\mathcal{R}$, éléments du treillis $\mathcal{L}(\mathcal{R})$, sont appelés des* ***concepts formels*** *ou plus simplement* ***concepts****.*

Pour nos études de complexité, nous utiliserons les notations $n = |\mathscr{P}|+|\mathscr{O}|$ et $m = |\bar{\mathcal{R}}|$; pour simplifier, nous assimilerons souvent $|\mathscr{P}|$ à n.

En l'absence de notation standard, nous avons choisi d'utiliser dans nos exemples des **lettres** pour représenter les propriétés (éléments de $\mathscr{P}$) et des **nombres** pour représenter les objets (éléments de $\mathscr{O}$).

Il n'y a pas non plus de convention qui se dégage sur le mode de construction d'un treillis des concepts : on peut le définir comme le treillis de Galois d'une relation sur $(\mathscr{P}, \mathscr{O})$ (construction par inclusion sur les propriétés) ou faire le choix dual d'une relation $\mathcal{R}$ sur $(\mathscr{O}, \mathscr{P})$ (inclusion sur les objets). Avec le premier choix, un concept $A \times B$ aura pour membre gauche un ensemble A de propriétés et pour membre droit un ensemble B d'objets ; avec le second choix, A sera un ensemble d'objets et B un ensemble de propriétés.

Les notions d'**intension** et d'**extension** sont par contre clairement définies : l'intension d'un concept est un ensemble de propriétés, l'extension d'un concept est un ensemble d'objets.

Nous avons pris comme convention le premier choix où les **propriétés** sont dans le membre **gauche** d'un concept (intension à gauche) et les objets dans le membre droit (extension à droite). Ainsi, $\perp = A \times \mathscr{O}$ et $\top = \mathscr{P} \times B$, avec $A = B = \emptyset$ si la relation ne contient aucune rangée de uns.

Nous appellerons **relation complète** la relation $\mathcal{R} = \mathscr{P} \times \mathscr{O}$; son treillis des concepts est réduit à un élément : $\mathscr{P} \times \mathscr{O}$. Nous appellerons **relation vide** la relation $\mathcal{R} = \emptyset$; son treillis des concepts a deux éléments $\perp = \emptyset \times \mathscr{O}$ et $\top = \mathscr{P} \times \emptyset$.

Par construction, un treillis des concepts a une hauteur bornée par $h = min(|\mathscr{P}|, |\mathscr{O}|)$; la longueur d'une chaîne maximale y sera donc inférieure ou égale à h. Le nombre de concepts peut pourtant être exponentiel (en $O(2^n)$) car le treillis peut être très large (certaines antichaînes maximales peuvent avoir des tailles exponentielles). Le pire cas correspond à un treillis des parties (*power set*) sur $(\mathscr{P}, \mathscr{O})$ qui est caractérisé par le fait que pour toutes les parties $A \subseteq \mathscr{P}$ et $B \subseteq \mathscr{O}$, l'élément $A \times B$ appartient au treillis.

Exemple 1.2.51
Soit un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ avec $\mathscr{P} = \{a, b, c, d, e, f\}$, $\mathscr{O} = \{1, 2, 3, 4, 5, 6\}$ et $\mathcal{R}$ la relation définie par la table ci-dessous :

$\mathcal{R}$	a	b	c	d	e	f
1		x	x	x	x	
2	x	x	x			
3	x	x				x
4				x	x	
5			x	x		
6	x					

Le treillis des concepts $\mathcal{L}(\mathcal{R})$ associé à $\mathcal{R}$ est représenté dans la figure 1.1.
La relation $\mathcal{R}$ n'a pas de ligne de uns, mais elle n'est pas réduite car la colonne f est

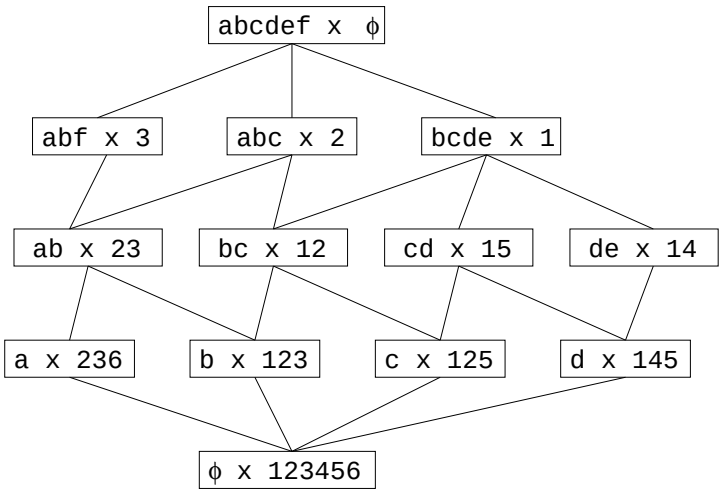


FIG. 1.1 – Treillis des concepts $\mathcal{L}(\mathcal{R})$ associé à la relation de l'exemple 1.2.51

l'intersection des colonnes a et b.
$bc \times 12$ et $abf \times 3$ sont des rectangles maximaux (concepts) de $\mathcal{R}$ et des éléments de $\mathcal{L}(\mathcal{R})$.
bc est l'intension de $bc \times 12$, 12 est son extension.
$ab \times 23$ et $bc \times 12$ ne sont pas comparables, $ab \times 23$ est un successeur de $a \times 236$, $a \times 236$ est un prédécesseur de $ab \times 23$.
Les atomes de $\mathcal{L}(\mathcal{R})$ sont : $a \times 236$, $b \times 123$, $c \times 125$ et $d \times 145$.
$(\emptyset \times 123456$, $b \times 123$, $bc \times 12$, $abc \times 2$, $abcdef \times \emptyset)$ est une chaîne maximale de $\mathcal{L}(\mathcal{R})$.

Cet exemple sera repris dans le chapitre 4.
◇

1.3 Graphes non-orientés

Nous rappelons ici les notions de graphes non orientés dont nous aurons besoin pour notre étude : après les principales définitions et notations et la présentation de quelques classes de graphes, nous abordons la notion de séparateur minimal dans ses aspects formels puis algorithmiques, nous terminons par la triangulation minimale d'un graphe. Pour plus de précisions sur les graphes, le lecteur pourra se référer aux livres de Golumbic [Gol80] et de Brandstädt & al. [BLS99].

1.3.1 Définitions et notations

Définition 1.3.1 GRAPHE NON-ORIENTÉ.
Un graphe ***non-orienté*** $G = (V, E)$ *est défini par la donnée d'un ensemble* V *de sommets* ($n = |V|$) *et d'un ensemble* E ($m = |E|$) *d'****arêtes*** *qui sont des paires* $\{x, y\}$ *de sommets de* V *(notées* xy*). Nous dirons que* x ***voit*** y *(et réciproquement) ou encore que* x *et* y *sont* ***voisins*** *dans* G *si* $xy \in E$*. Le* ***voisinage d'un sommet*** x *est l'ensemble, noté* $N(x)$*, des voisins de* x *(*x *non compris). Le* ***voisinage d'un ensemble*** X *de sommets est l'ensemble*

de ses **voisins extérieurs** *:* $N(X) = \bigcup_{x \in X} N(x) - X$. *Le* ***degré*** *d'un sommet* x *est la taille* $|N(x)|$ *de son voisinage.*

Propriété 1.3.2
Tout graphe orienté $G = (V, A)$ *est associé à un unique graphe non-orienté* $G' = (V, E)$ *obtenu par fermeture symétrique de* G *:* $xy \in E \iff xy \in A \vee yx \in A$.

On définit donc la matrice d'incidence $G = (V, E)$ d'un graphe non-orienté sur le même principe que la matrice d'incidence d'un graphe orienté comme un tableau booléen M à deux indices tel que $\forall x, y \in V$, $M[x, y] = 1 \iff xy \in A$. Cette matrice est symétrique.

D'autre part, le parcours d'un graphe non orienté se fera sur le même principe que celui des graphes orientés (page 1.1.19), l'ensemble des sommets atteints à partir d'un sommet donné correspondant à sa composante connexe.

Par la suite, le terme **graphe** fera référence à un graphe *non-orienté*. Nous noterons $G(X)$ le sous-graphe de $G = (V, E)$ **induit** par $X \subseteq V$ c'est-à-dire ayant X pour ensemble de sommets et pour ensemble d'arêtes {xy∈E | x,y∈X} ; nous dirons que $G(X)$ **est inclus dans** G (inclusion au sens des arêtes). Pour simplifier les notations, nous écrirons souvent X au lieu de $G(X)$ quand il n'y a pas d'ambiguïté.

Définition 1.3.3 CLIQUE, STABLE ; SATURATION.
Une ***clique*** *est un graphe complet, c'est-à-dire un graphe* $G = (V, E)$ *possédant toutes les arêtes :* $E = \{xy \mid x, y \in V\}$. *Un* ***stable*** *(independent set) est un graphe sans arête. Nous dirons qu'on* ***sature*** *un ensemble* $X \subseteq V$ *de sommets quand on ajoute toutes les arêtes nécessaires pour que* $G(X)$ *soit une clique.*

Dans un graphe $G = (V, E)$, nous appellerons **chemin** toute suite $(x_1, ..., x_k)$ de sommets distincts tels que $\forall i \in [2..k]$, x_{i-1} et x_i sont voisins ; un tel chemin forme un **cycle** quand x_k et x_1 sont voisins. Une **corde** dans un cycle est une arête entre deux sommets non consécutifs, c'est-à-dire une arête $x_i x_j$ avec $|i-j| \notin \{1, k-1\}$.

Définition 1.3.4 CONNEXITÉ.
Un graphe est dit ***connexe*** *si, pour toute paire de sommets du graphe, il existe un chemin qui les relie. Une* ***composante connexe*** *d'un graphe est un sous-graphe connexe non vide, maximal au sens du nombre de sommets.*

Un graphe non connexe est donc formé d'au moins deux composantes connexes non vides.

Un sommet est dit **isolé** si son voisinage est vide. Un sommet est dit **universel** s'il voit tous les autres sommets du graphe. Un sommet non isolé est dit **simplicial** si son voisinage est une clique.

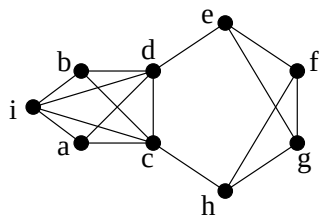
Définition 1.3.5 MODULE.
On appelle ***module*** *(homogeneous set) d'un graphe G=(V,E), toute partie propre* X *de* V *dont tous les sommets ont même voisinage extérieur (i.e.* $\forall x, y \in X$, $N(x) - X = N(y) - X$*).*

Un module X est dit ***complet*** *si G(X) est une clique ; il est dit* ***complet maximal*** *si de plus aucune partie de V incluant strictement X n'est un module complet. Dans la suite de ce mémoire un module complet maximal sera dénommé* **maxmod**.

Un sous-ensemble X de sommets sera donc un module complet si et seulement si $\forall x, y \in X$, $N(x) \cup \{x\} = N(y) \cup \{y\}$.

Théorème 1.3.6 *([BB98]).*
*La relation "*appartient à un même maxmod*" définit entre les sommets d'un graphe une relation d'équivalence.*

Exemple 1.3.7
Un graphe non orienté connexe $G = (V, E)$.
$V = \{a, b, c, d, e, f, g, h, i\}$ et $E = \{ac, ad, ai, bc, bd, bi, cd, ch, ci, de, ef, eg, fg, fh, gh\}$.



Le graphe G

a est simplicial : $N(a) = \{c, d, i\}$ est une clique. $\{a, c, d, i\}$ et $\{e, f, g\}$ sont des cliques maximales. $\{a, b, e, h\}$ est un stable. $cdefghc$ est un cycle avec deux cordes fh et eg ; $cdegh$ est un cycle sans corde. $\{f, g\}$ est un maxmod : $N(f) \cup \{f\} = \{e, f, g, h\} = N(g) \cup \{g\}$.
$\diamond$

On verra dans la suite qu'il sera utile de contracter les sommets d'un maxmod. De façon général la **contraction de deux sommets** x et y dans un graphe G est définie comme suit : remplacer x et y par un sommet z tel que $N(z) = N(x) \cup N(y)$.

On verra aussi que nous utilisons souvent des **schémas d'élimination**, qu'il s'agisse d'élimination de sommets ou d'arêtes. Un schéma d'élimination **par sommets** consiste à itérativement supprimer un sommet du graphe et les arêtes incidentes ; un schéma d'élimination **par arêtes** consiste à itérativement supprimer une arête du graphe. On définit dualement les **schémas de composition** par arêtes ou par sommets.

1.3.2 Classes de graphes

Définition 1.3.8 ARBRE.
Un arbre est un graphe connexe et sans cycle.

Caractérisation 1.3.9 d'un arbre
Un graphe G est un arbre si et seulement si une des propriétés suivantes est vérifiée :

- *G est sans cycle et possède n sommets et $m = n-1$ arcs ;*

- *G est connexe et possède n sommets et $m = n-1$ arcs ;*
- *G est connexe et on le déconnecte en retirant une arête quelconque.*
- *G est sans cycle et on crée un unique cycle en ajoutant une arête quelconque.*
- *Pour toute paire $\{x, y\}$ de sommets de G, il existe un chemin unique d'extrémités x et y.*

Théorème 1.3.10
On peut orienter d'une manière unique un arbre en arborescence en choisissant une racine parmi n'importe lequel de ses sommets.

Les arborescences sont ainsi souvent appelées **arbres enracinés**.

Définition 1.3.11 GRAPHE BIPARTI *(non orienté).*
*Un graphe $G = (V, E)$ est dit **biparti** s'il existe une partition de V en X et Y tels que $G(X)$ et $G(Y)$ soient des stables.*
*Un graphe $G = (V, E)$ est dit **co-biparti** si son complémentaire $\bar{G} = (V, V^2 - E)$ est biparti.*

La version orientée de la classe des graphes bipartis a été présentée dans la définition 1.1.14. Dans un graphe non-orienté co-biparti, il existe donc une partition des sommets en deux cliques, avec un certain nombre d'arêtes reliant ces deux cliques.

Définition 1.3.12 GRAPHE TRIANGULÉ.
*Un graphe G=(V,E) est dit **triangulé**, ou encore **cordal** (*triangulated, chordal*), s'il ne possède aucun cycle sans corde.*

La reconnaissance algorithmique d'un graphe triangulé se fait en temps linéaire à l'aide de l'algorithme `Lex-BFS` (voir [RTL76]) ou de l'algorithme `MCS` (voir [TY84]). Ces deux algorithmes sont basés sur un parcours du graphe où le choix du prochain sommet à atteindre est déterminé par une MARQUE. Les sommets atteints sont numérotés par ordre décroissant n à 1, le premier sommet de l'ordre total obtenu étant atteint en dernier.

Dans l'algorithme `Lex-BFS`, la MARQUE de chaque sommet est une suite d'entiers. Les suites sont comparées en utilisant l'ordre lexicographique inverse, par exemple : $(7) \geq (6, 2) \geq (5, 4, 2) \geq (5, 4) \geq ()$.

Algorithme 1.3.13 `Lex-BFS` ([RTL76])
Donnée : Un graphe $G = (V, E)$.
Résultat : Une numérotation NUM ordonnant les sommets de G.
Initialisation
ATTEINT $\leftarrow \emptyset$;
$i \leftarrow 0$;
Pour $x \in V$ **faire :** MARQUE$(x) \leftarrow ()$;
Début
Tant que ATTEINT$\neq V$ **faire :**
  Choisir $x \in V-$ATTEINT tel que MARQUE(x) soit maximal ;

Pour $y \in N(x)-$ATTEINT **faire :**
Ajouter la valeur $i+1$ à la fin de la liste MARQUE(y) ;
NUM$(x) \leftarrow n-i$;
$i \leftarrow i+1$;
Fin.

Dans l'algorithme `MCS`, la MARQUE de chaque sommet est un entier dont la valeur est comprise entre 0 et n.

Algorithme 1.3.14 `MCS` ([TY84])
Donnée : Un graphe $G = (V, E)$.
Résultat : Une numérotation NUM ordonnant les sommets de G.
Initialisation
ATTEINT $\leftarrow \emptyset$;
$i \leftarrow 0$;
Pour $x \in V$ **faire :** MARQUE$(x) \leftarrow 0$;
Début
Tant que ATTEINT$\neq V$ **faire :**
Choisir $x \in V-$ATTEINT tel que MARQUE(x) soit maximal ;
Pour $y \in N(x)-$ATTEINT **faire :**
MARQUE$(y) \leftarrow$ MARQUE$(y) + 1$;
NUM$(x) \leftarrow n-i$;
$i \leftarrow i+1$;
Fin.

Définition 1.3.15 ORDRE D'ÉLIMINATION PARFAIT
Un ordre total $(x_1, ..., x_n)$ sur les sommets d'un graphe $G = (V, E)$ est appelé ordre d'élimination parfait (perfect elimination ordering) quand, pour $i \in [1..n]$, le sommet x_i est simplicial dans $G(x_i, ..., x_n)$.

Théorème 1.3.16 *([RTL76], [TY84])*
Un graphe est triangulé si et seulement si l'ordre sur les sommets fourni par une exécution de `MCS` *ou de* `Lex-BFS` *est un ordre d'élimination parfait.*

Ainsi, pour vérifier si un graphe est triangulé, on procède en deux temps :

1. Une exécution de `MCS` ou de `Lex-BFS` fournit, en temps linéaire, un ordre d'élimination parfait $(x_1, ..., x_n)$;
2. On vérifie, en temps linéaire, que, pour i allant de 1 à n, le sommet x_i est simplicial dans $G(x_i, ..., x_n)$.

Définition 1.3.17 GRAPHE FAIBLEMENT TRIANGULÉ *([Hay85]).*
Un ***trou*** *dans un graphe est un cycle de longueur supérieure à quatre sans corde. Un graphe est dit* **sans trou** *(*hole-free*) quand il n'a aucun trou.*
Un graphe G est dit ***faiblement triangulé*** *quand G et $\bar{G}$ sont sans trou.*

Hayward, qui a défini cette classe de graphes, a proposé pour elle un schéma de composition incrémental par arêtes (voir [Hay96]). Il existe plusieurs algorithmes en $O(m^2)$ de

reconnaissance (voir [HSS00], [BBH00]). Ces graphes présentent des propriétés similaires à celles des graphes triangulés (voir [BBH00], [BB00]) qu'ils généralisent.

Nous aurons encore besoin de quelques classes de graphes.

Définition 1.3.18 GRAPHES SANS TRIPLET ASTÉROÏDAL (AT-FREE) *([LB62]).*
Dans un graphe, un triplet $\{x_1, x_2, x_3\}$ de sommets formant un stable est dit ***triplet astéroïdal*** *quand, pour chaque paire $\{x_i, x_j\}$ issu du triplet, il existe un chemin entre x_i et x_j qui ne passe par le voisinage du troisième sommet x_k.*
Un graphe est dit **AT-free** *(ATF,* sans triplet astéroïdal*) quand il ne possède aucun triplet astéroïdal.*

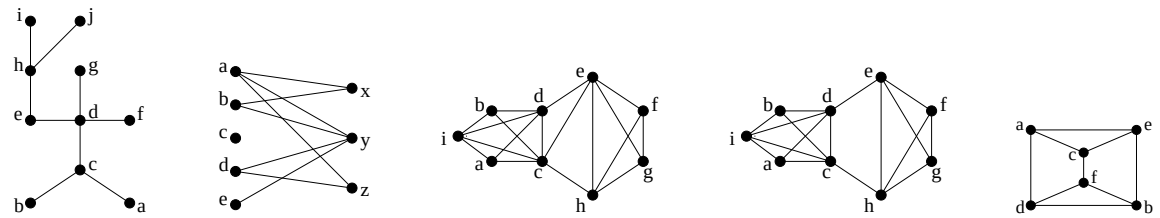
Définition 1.3.19 GRAPHES SANS GRIFFE (CLAW-FREE).
On appelle ***claw*** *(litt.* griffe*) un graphe isomorphe à $K_{1,3}$.*
Un graphe est dit ***Claw-free*** *quand il n'admet pas $K_{1,3}$ comme sous-graphe.*

Définition 1.3.20 GRAPHES D'INTERVALLES.
Un graphe $G = (V, E)$ est un graphe d'intervalles s'il existe une bijection φ de V vers un ensemble $\mathcal{I}$ d'intervalles de $I\!R$ telle que $xy \in E \iff \varphi(x) \cap \varphi(y) \neq \emptyset$.

Définition 1.3.21 GRAPHE D'INTERVALLES PROPRES.
Un graphe d'intervalles est dit graphe d'intervalles propres quand il peut être défini avec un ensemble d'intervalles dans lequel aucun intervalle n'est inclus proprement dans un autre.

Pour plus de précisions sur les graphes d'intervalles propres, se référer à [LB62], [Rob69] et [Gol80].

Exemple 1.3.22
Quelques exemples de graphes connexes..



G_1 G_2 G_3 G_4 G_5

G_1 est un arbre. G_2 est un graphe biparti ; il correspond à la fermeture symétrique du graphe orienté G de l'exemple 1.1.16. G_3 est triangulé. G_4 est faiblement triangulé ; il n'est pas triangulé car $cdehc$ est un cycle de taille 4 sans corde. G_5 est un graphe sans trou ; il n'est pas faiblement triangulé car $abcdefa$ est un cycle sans corde de son complémentaire.

Le graphe G de l'exemple 1.3.7 a pour trou $cdeghc$ et n'est donc pas faiblement triangulé, ni triangulé.

◇

1.3.3 Séparateurs minimaux

La notion de séparateur minimal a été introduite par Dirac en 1961 ([Dir61]) pour caractériser la famille des graphes triangulés. Généralisée à des graphes quelconques, la notion a été très étudiée dans la dernière décennie avec les travaux de Kloks, Kratsch et Spinrad ([KKS93] et [KK98]), de Parra et Schefler ([PS95] et [Par96]), de Todinca ([Tod99]), ainsi que de Berry et Bordat (voir notamment [Ber98] et [BBC00]).

La puissance de ce nouvel outil d'étude des graphes a fourni des résulats théoriques et algorithmiques majeurs pour la décomposition d'un graphe (voir la sous-section suivante).

Un certain nombre de propriétés sur la séparation sont utilisées sans que la preuve en ait jamais été publiée. Nous présentons donc ici les preuves des propriétés utilisées quand elles ne figurent pas explicitement dans une publication ou quand – à notre connaissance – ces propriétés sont nouvelles.

Définition 1.3.23 SÉPARATEURS.
• *On appelle* ***séparateur*** *dans un graphe connexe $G = (V, E)$ toute partie propre S de V dont le retrait déconnecte le graphe (autrement dit, $G(V-S)$ comprend au moins deux composantes connexes).*
• *Un séparateur S est dit* ***ab-séparateur*** *si dans $G(V-S)$ les sommets a et b se trouvent dans deux composantes connexes distinctes.*
• *Un* ***ab-séparateur*** *S est dit* ***minimal*** *si aucun sous-ensemble propre de S n'est un ab-séparateur.*
• *Un* ***séparateur*** *S est dit* ***minimal*** *s'il existe deux sommets a et b tels que S soit un ab-séparateur minimal.*

Un séparateur permet donc de partitionner les sommets du graphe en ce séparateur et les différentes composantes connexes que son retrait induit.

La définition d'un séparateur peut se généraliser aux graphes non-connexes en considérant comme séparateur d'un graphe non-connexe tout séparateur d'une composante connexe du graphe. La séparation dans un graphe non connexe nous ramène ainsi à la séparation dans un graphe connexe, en raisonnant sur la composante à déconnecter au lieu de raisonner sur tout le graphe.

La définition de la séparation se traduit en termes de chemin comme suit :

Caractérisation 1.3.24
Soient $G = (V, E)$ un graphe, $S \subset V$ et $x, y \in V-S$.
- *S est un xy-séparateur de G si et seulement si tout chemin reliant x et y passe par au moins un sommet de S ;*
- *De plus, S est minimal si et seulement si tout sommet de S appartient à un chemin reliant x et y et ne passant par aucun autre sommet de S.*

La possibilité de réduire un séparateur jusqu'à obtenir un séparateur minimal est garantie par le corollaire suivant.

Corollaire 1.3.25
Soient $G = (V, E)$ un graphe, x et y deux sommets de G et $S \subset V$. Si S est un xy-séparateur alors il existe un xy-séparateur ***minimal*** *$T \subseteq S$.*

Preuve : Soit S est lui-même xy-séparateur minimal ; soit ce n'est pas le cas et, par la caractérisation 1.3.24, il existe un sommet z de S qui n'est sur aucun chemin reliant x et y sans passer par un autre sommet de S. $S-\{z\}$ est alors un xy-séparateur. On peut retirer de S tous les sommets qui partagent cette propriété avec z, et obtenir ainsi un xy-séparateur minimal.
◇

Corollaire 1.3.26
Soient $G = (V, E)$ un graphe connexe. Si x et y sont deux sommets non voisins de G, alors G possède un xy-séparateur minimal.

Preuve : $G(\{x, y\})$ n'est pas connexe puisque x et y ne sont pas voisins. $V-\{x, y\}$ est non vide puisque G est connexe et qu'il y a donc un chemin entre x et y qui passe par au moins un autre sommet. Par conséquent, le retrait de $V-\{x, y\}$ déconnecte G en au moins deux composantes connexes contenant respectivement x et y ; donc $V-\{x, y\}$ est un xy-séparateur. Le corollaire 1.3.25 permet d'affirmer alors que G possède un xy-séparateur minimal.
◇

Il est important de noter qu'un séparateur qui sera minimal pour la séparation d'une paire xy de sommets donnée ne sera pas forcément minimal pour la séparation d'une autre paire ab, ni même séparateur pour cette autre paire.

Définition 1.3.27 COMPOSANTE PLEINE.
Soit S un séparateur d'un graphe $G = (V, E)$ et soit C une composante connexe de $G(V-S)$. C est dite ***composante pleine*** *pour S quand $N(C) = S$.*

Caractérisation 1.3.28 d'un séparateur minimal.
Soit $G = (V, E)$ un graphe. Un séparateur $S \subset V$ de G est un séparateur minimal si et seulement si $G(V-S)$ admet au moins deux composantes connexes pleines.

Cette propriété, utilisée depuis des années, notamment dans [Ber98], est difficile à attribuer à un auteur particulier ; nous en donnons une preuve ci-dessous.

Preuve : Soit G un graphe.

$\Longrightarrow$

Soient a et b deux sommets de G et S un ab-séparateur minimal. Supposons que S induise au plus une composante pleine. Par définition d'un séparateur, il existe une composante C_1 induite par S qui n'est pas pleine. On peut supposer, sans perte de généralité, que $a \in C_1$. Comme C_1 n'est pas pleine, il existe dans S un sommet x qui ne voit pas C_1 ; en particulier, l'arête xa est absente. Donc $S-\{x\}$ est aussi ab-séparateur, ainsi S n'est pas minimal pour la séparation de a et b, ce qui est en contradiction avec l'hypothèse.

$\Longleftarrow$

Soit S un séparateur de G qui induit au moins deux composantes pleines C_1 et C_2. Soient $a \in C_1$ et $b \in C_2$. S est un ab-séparateur minimal ; en effet, soit un sommet x quelconque de S. $S - \{x\}$ ne sépare pas a de b car il y a un chemin entre a et x (puisque C_1 est composante pleine) et un chemin entre b et x.

$\diamond$

Nous pouvons en déduire le corollaire suivant, qui nous sera utile pour certaines preuves.

Corollaire 1.3.29
Soient G un graphe, S un séparateur minimal induisant les composantes connexes pleines C_1 et C_2. Tout sommet de S a un voisin dans C_1 et un voisin dans C_2, lesquels voisins ne se voient pas.

Preuve : Soient G un graphe, S un séparateur minimal induisant les composantes connexes pleines C_1 et C_2.
Comme C_1 et C_2 sont deux composantes pleines, elles vérifient $N(C_1) = N(C_2) = S$; donc tout sommet de S est voisin d'un sommet de C_1 et d'un sommet de C_2. Soit $x \in S$ ayant notamment pour voisins y_1 dans C_1 et y_2 dans C_2 ; puisque S est un séparateur, y_1 et y_2 ne peuvent être voisins.

$\diamond$

Exemple 1.3.30
Séparateurs dans le graphe G de l'exemple 1.3.7.
cdi est un séparateur minimal complet car il est minimal pour la séparation de a et b et il est complet. cdi induit les composantes pleines $\{a\}$, $\{b\}$ et la composante non pleine $\{e, f, g, h\}$ (i ne voit aucun sommet de cette composante). cdi sépare aussi a de h, mais il n'est pas minimal pour cette séparation car cd suffit pour séparer a et h.
$\{d, h\}$ est un séparateur minimal non complet.

$\diamond$

Pour terminer ces rappels sur les séparateurs minimaux, nous présentons maintenant la notion de séparateurs minimaux croisants, qui sera utilisée au chapitre 4.

Définition 1.3.31 *([KKS93]).*
Soit S et T deux séparateurs minimaux d'un graphe G. On dit que T ***croise*** *S quand il y a dans $G(V-S)$ deux composantes connexes différentes C_1 et C_2 telles que $T \cap C_1 \neq \emptyset$ et $T \cap C_2 \neq \emptyset$.*

Caractérisation 1.3.32 *([Par96]).*
Un séparateur minimal d'un graphe G est complet si et seulement si il ne croise aucun autre séparateur minimal de G.

Propriété 1.3.33 *([PS95]).*
Soient G un graphe, S un séparateur minimal de G, et G_S le graphe obtenu en saturant

S dans G. T est un séparateur minimal de G_S si et seulement si T est un séparateur minimal de G qui ne croise pas S dans G.

Il a été montré que cette relation est symétrique (voir [PS95]) : si S croise T alors T croise S ; nous dirons donc que S et T sont (ou ne sont pas) croisants.

Remarque 1.3.34 *La notation G_S, avec S séparateur minimal est consacrée par l'usage, elle ne devra pas être confondue avec l'écriture $G_{\mathcal{R}}$ du graphe sous-jacent à une relation $\mathcal{R}$ défini dans le chapitre 4.*

Exemple 1.3.35
Séparateurs minimaux croisants dans le graphe G de l'exemple 1.3.7.
Le séparateur minimal non complet $\{d, h\}$ induit les composantes $\{a, b, c, i\}$ et $\{e, f, g\}$. Il croise le séparateur minimal non complet $\{c, e\}$ qui induit les composantes $\{a, b, d, i\}$ et $\{f, g, h\}$. En effet, $\{d, h\} \cap \{a, b, d, i\} = \{d\} \neq \emptyset$ et $\{d, h\} \cap \{f, g, h\} = \{h\} \neq \emptyset$.
La saturation de $\{d, h\}$ créerait par exemple le nouveau chemin $adhg$ qui empêcherait la séparation par $\{c, e\}$ de a et g ; dans $G-\{dh\}$, $\{c, e\}$ n'est plus séparateur minimal.
$\{cd\}$ est un séparateur minimal complet, il ne croise donc aucun séparateur minimal.
$\diamond$

1.3.4 Décomposition par séparateurs minimaux complets

C'est Tarjan (voir [Tar85]) qui a, le premier, utilisé les séparateurs complets pour la décomposition d'un graphe quelconque.

Définition 1.3.36 DÉCOMPOSITION PAR SÉPARATEUR COMPLET *([Tar85]).*
Soit $G = (V, E)$ un graphe. Soient S un séparateur complet de G et C_1,...,C_k les composantes connexes de $G(V-S)$. La décomposition de G par S consiste à remplacer G par les sous-graphes $G(C_1 \cup S)$,..., $G(C_k \cup S)$.

Il a été prouvé que ce type de décomposition est unique et optimale (décomposition en sous-graphes de taille maximale mais en nombre minimal) quand seuls des séparateurs minimaux complets sont utilisés (voir [Lei93]) et on peut le définir par l'étape général de décomposition suivante (voir [BB97]) :

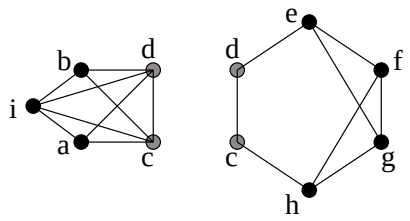
Etape de décomposition 1.3.37 par séparateur minimal complet :

Donnée : un graphe connexe G, un séparateur minimal complet S de G qui définit k composantes connexes C_1, ..., C_k.
Résultat : une décomposition $G_1, ..., G_k$ de G.
Etape :
Remplacer G par les graphes $G_1 = G(C_1 \cup N(C_1))$, ..., $G_k = G(C_k \cup N(C_k))$.

Cette décomposition a la propriété remarquable de partitionner les séparateurs minimaux de G entre les sous-graphes qu'elle définit :

Propriété 1.3.38 *([BB97]).*
Soit G un graphe, $\mathscr{S}(G)$ l'ensemble de ses séparateurs minimaux et S un séparateur minimal de G. Après une étape de décomposition 1.3.37 de G par S, $\mathscr{S}(G)-\{S\}$ est partitionné dans les différents sous-graphes obtenus.

Exemple 1.3.39
La décomposition du graphe G de l'exemple 1.3.7 (page 36) par le séparateur minimal complet $\{c, d\}$, aboutit à deux sous graphes $G(\{c, d, e, f, g, h\})$ et $G(\{a, b, c, d, i\})$.



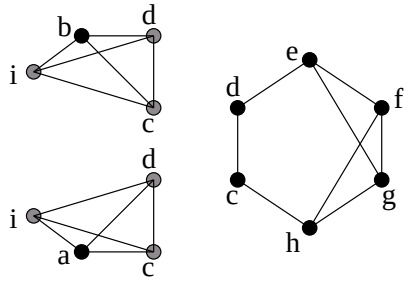
Décomposition de G par $\{c, d\}$

◇

L'étape de décomposition par séparateurs minimaux complets peut être répétée sur les sous-graphes obtenus jusqu'à stabilité :

Schéma de décomposition 1.3.40 par séparateurs minimaux complets.
Donnée : un graphe connexe G et l'ensemble $\mathscr{S}$ de ses séparateurs minimaux complets (les séparateurs minimaux complets sont notés s.m.c.).
Résultat : la décomposition de G par séparateurs minimaux complets.
Schéma :
Tant que il reste un graphe comportant des s.m.c. **faire :**
Réaliser une étape de décomposition 1.3.37 sur ce graphe par un de ses s.m.c. ;
Remplacer ce graphe par les sous-graphes obtenus.

Exemple 1.3.41
La décomposition du graphe G de l'exemple 1.3.39 peut se poursuivre : le séparateur minimal complet $\{c, d, i\}$ décompose $G(\{a, b, c, d, i\})$ en $G(\{a, c, d, i\})$ et $G(\{b, c, d, i\})$.
Les différents sous-graphes obtenus ne contiennent plus de séparateur minimal complet, la décomposition s'arrête.



Le résultat de la décomposition de G, de l'exemple 1.3.7, par $\{c, d\}$ puis $\{c, d, i\}$.

◇

Le schéma de décomposition 1.3.40 est implémentable en $O(nm)$. Cette complexité est rendue possible par la propriété remarquable suivante :

Propriété 1.3.42 *([PS95]).*
Un séparateur minimal d'un graphe connexe G est un séparateur minimal complet si et seulement si il est un séparateur minimal de toute triangulation minimale de G.

Pour obtenir en $O(nm)$ l'ensemble des séparateurs minimaux complets d'un graphe G, en s'appuyant sur le procédé de triangulation minimale défini dans la sous-section suivante, il suffit donc de :

1. Calculer, en $O(nm)$, une triangulation minimale quelconque H de G ;
2. Calculer les séparateurs minimaux de H, ce qui se fait en $O(m')$, où m' est le nombre d'arêtes dans H ;
3. Tester, en $O(nm)$, parmi les séparateurs minimaux de H ceux qui étaient déjà complets dans G.

Ce procédé peut être étendu aux séparateurs minimaux non complets si on sature un séparateur minimal avant l'étape de décomposition. La saturation d'un séparateur minimal provoque la disparition dans le graphe d'un certain nombre d'autres séparateurs minimaux ; ce procédé a été introduit pour la première fois par [KKS93] pour utiliser les séparateurs minimaux d'un graphe afin de calculer une triangulation minimale de ce graphe. Il a été étudié extensivement dans [Ber98] et [BB97].

1.3.5 Triangulation minimale

La triangulation est un procédé algorithmique consistant à ajouter un certain nombre d'arêtes à un graphe afin d'obtenir un graphe triangulé dont le graphe de départ est graphe partiel. Il serait bien évidemment intéressant de rechercher le *plus petit nombre d'arêtes* qu'il faudrait ajouter à un graphe pour obtenir un graphe triangulé. Ce procédé est appelé triangulation **minimum**. La recherche d'une triangulation minimum d'un graphe est malheureusement un problème NP-Complet (voir [TY84]). Par contre le procédé consistant à ajouter une quantité d'arêtes juste suffisante pour obtenir un graphe triangulé, sans considération sur la minimalité, est algorithmiquement plus accessible. Ce procédé, et le graphe triangulé qui résulte de son application est appelé triangulation **minimale**. La définition de ce concept est due à Ohtsuki, Cheung et Fujisawa ([OCF76]) et a été étudié par Rose, Tarjan et Lueker :

Définition 1.3.43 TRIANGULATION MINIMALE *([RTL76]).*
Soit $G = (V, E)$ un graphe non triangulé. On appelle triangulation minimale de G tout graphe $H = (V, E+F)$, tel que pour toute partie propre F' de F, le graphe $H' = (V, E+F')$ n'est pas triangulé.

Ils ont aussi démontré dans le même papier qu'il suffit de retirer une arête à H pour vérifier qu'il est bien une triangulation minimale de G :

Définition 1.3.44 *([RTL76]).*
Soit $G = (V, E)$ un graphe quelconque et $H = (V, E + F)$ un graphe triangulé. H est une triangulation minimale de G si et seulement si pour toute arête $f \in F$ le graphe $(V, E+(F-\{f\}))$ n'est pas triangulé.

La triangulation minimale a été bien étudiée (voir [Ber99], [BHT01], [OCF76], [RTL76], [TY84]). En particulier, des travaux récents ont montré que les séparateurs minimaux peuvent être utilisés pour calculer une triangulation minimale (voir [KKS93], [PS95], [Ber98]).

Exemple 1.3.45
Le graphe G_3 de l'exemple 1.3.22 est un graphe triangulé qui peut être obtenu à partir du graphe G de l'exemple 1.3.7 par saturation des séparateurs minimaux non complets $\{c, e\}$ et $\{e, h\}$.
$\diamond$

La meilleure complexité de calcul de triangulation minimale, $O(nm)$, peut être obtenue en utilisant un des trois algorithmes conçus spécifiquement : `LEX-M` ([RTL76]), `LB-TRIANG` ([Ber99]) ou `MCSM` ([BBH00]). Nous rappelons ici l'algorithme `LB-TRIANG` qui présente l'avantage d'utiliser, sur les sommets du graphe initial, un ordre α qui peut être imposé par l'utilisateur.

Algorithme 1.3.46 `LB-TRIANG` ([Ber99]).

Donnée : Un graphe $G = (V, E)$, un ordre α sur V.
Résultat : Un ensemble F d'arêtes et une triangulation minimale $H = (V, E+F)$ de G.
Initialisation :
  $H \leftarrow G$;
  $F \leftarrow \emptyset$;
Début
Pour chaque sommet x de V pris dans l'ordre α **faire :**
  Pour chaque composante connexe C de $G(V-(N_H(x) \cup \{x\}))$ **faire :**
    $F' \leftarrow$ ensemble des arêtes à ajouter à $N_G(C)$ pour en faire une clique ;
    $F \leftarrow F + F'$;
    $H \leftarrow (V, E + F)$;
Fin.

Première partie

PHYLOGENIE

Introduction de la première partie

Ce n'est pas chose facile, pour un mathématicien ou un informaticien, que d'étudier un problème de biologie. Du fait de la diversité et de l'immensité du monde vivant, l'établissement d'un modèle abstrait est un processus lent : de nombreuses connaissances doivent être intégrées et de nombreuses expériences réalisées, avec le problème supplémentaire posé par les inévitables incertitudes dans les résultats expérimentaux obtenus. Les modèles mathématiques des problèmes biologiques sont ainsi souvent issus d'un consensus, établi entre mathématiciens et biologistes, chaque chercheur enrichissant sa vision du modèle de son expérience et de son intuition propres. Ce travail de va-et-vient prend du temps et la mise au point d'un tel modèle dépasse le seul cadre d'une thèse.

Comme tant d'autres, nous nous sommes trouvés confrontés à ces problèmes. Dans cette première partie, nous décrirons donc la première phase de ce travail de va-et-vient : à l'écoute d'un problème spécifique des biologistes, nous mettons au point une solution théorique sous la forme d'un algorithme. Les notions nouvelles de distance triangulée et de sous-triangulation maximale que nous établissons nous indiquent de nouvelles directions. Nous menons donc une deuxième étude théorique dans le but d'affiner encore notre méthode initiale.

Le problème que nous abordons ici est lié à l'établissement d'arbres d'évolution, appelés aussi *phylogénies*, à partir de mesures de dissimilarités. Notre approche consiste à utiliser les propriétés de symétrie de la matrice d'une distance additive d'arbre (et, plus généralement, d'une dissimilarité) pour définir des graphes, que nous étudions, et par le biais desquels nous traitons les données. Ainsi, nous transformons une telle matrice de valeurs en une famille de matrices booléennes, ce qui revient à un processus de binarisation. Nous proposons un algorithme d'ajustement d'une dissimilarité à une distance triangulée, pour ensuite procéder à une étude théorique complémentaire qui s'éloigne d'une problématique purement biologique.

Ce travail a été réalisé en collaboration avec Anne Berry et Christine Sinoquet, mais aussi avec François Barnabé ; nous avons ensemble découvert ce domaine de recherche qui présente des difficultés à la fois théoriques et expérimentales.

Les résultats de cette partie ont donné lieu à deux rapports de recherche :

- Le premier rapport [BSS02d] présente l'algorithme `ADD-SUB-TRI` qui transforme une distance quelconque en "distance triangulée". La notion nouvelle de sous-triangulation maximale, qui fonde l'algorithme, a été présentée en mai 2001 aux Journées Graphes et Algorithmes (JGA'01, Clermont-Ferrand).
- Le deuxième rapport [BSS02e] se propose d'améliorer `ADD-SUB-TRI` par une étude structurelle de la famille de graphes associée à une dissimilarité. Ces résultats ont été présentés en juin 2002, sous forme de Poster, au cours des Journées Ouvertes Biologie Informatique et Mathématiques (JOBIM'02, voir [BSS02f]). Les aspects graphiques ont été exposés en mars 2002 aux Journées Graphes et Algorithmes (JGA'02, Nantes).

Chapitre 2

Dissimilarité : interprétation par des graphes

Dans ce chapitre, nous donnerons quelques bases théoriques qui nous serviront pour les applications décrites dans le chapitre suivant. Nous commencerons par expliquer comment, à partir d'une dissimilarité, on définit une famille de graphes. Nous étudierons les familles de graphes associées à des dissimilarités particulières, rencontrées dans le cadre général de la classification arborée, et plus particulièrement dans celui de la phylogénie.

2.1 De la dissimilarité à la famille de graphes

Dans une perspective de taxonomie mathématique, quand on se propose de classer des objets, la première étape consiste à établir un critère de proximité ou d'éloignement entre ces objets. Si le critère est numérisable, une mesure de comparaison peut être établie expérimentalement en considérant l'ensemble d'objets et en mesurant l'éloignement des objets pris deux à deux. Ces données peuvent être représentées par une matrice symétrique à valeurs positives. Dans une matrice M, chaque case $M[i, j]$ contient la mesure obtenue en comparant l'objet i à l'objet j.

Une deuxième étape consiste à tenter de structurer ces données. Bien que notre approche soit très généralement applicable aux matrices de dissimilarité, nous nous intéressons ici à une problématique particulière, issue du domaine phylogénétique, qui est le point de départ de la nouvelle problématique de graphes que nous examinons.

Pour préciser mieux cette problématique, nous donnons dans cette section quelques rappels très généraux, essentiellement destinés aux lecteurs peu familiers du domaine. Nous définirons ensuite, dans la section suivante, la notion nouvelle de distance triangulée.

2.1.1 Dissimilarités, distances, représentations arborées

Nous travaillerons sur des matrices de dissimilarité :

Définition 2.1.1 DISSIMILARITÉ.
Soit X un ensemble. On appellera **dissimilarité sur** X *toute application δ de X^2 vers IR qui est :*

1. *Symétrique :* $\forall x, y \in X,\ \delta(x,y) = \delta(y,x)$, *et*
2. *Positive :* $\forall x, y \in X,\ \delta(x,y) \geq 0$.

Notons que certains auteurs (voir [BLM84]) utilisent l'appellation *indice de dissimilarité* au lieu de *dissimilarité.*

Définition 2.1.2 ECART, DISTANCE.
Une dissimilarité δ sur X peut avoir les propriétés suivantes :

(1) $\forall x \in X,\ \delta(x,x) = 0$ *;*

(2) "Intégrité" : $\forall x, y \in X,\ \delta(x,y) = 0 \Longrightarrow x = y$.

(3) Inégalité Triangulaire : $\forall x, y, z \in X,\ \delta(x,z) \leq \delta(x,y) + \delta(y,z)$.

Différentes définitions en découlent selon les propriétés que vérifie une dissimilarité :

(1) définit un ***indice d'écart****,*

(1)+(3) définit un ***écart****,*

(1)+(2) définit un ***indice de distance****, et enfin*

(1)+(2)+(3) définit une ***distance*** *(appelée aussi* ***métrique****).*

Nous nous pencherons plus particulièrement sur les matrices associées à des représentations dites arborées, pour lesquels il existe des caractérisations très intéressantes.

On sait qu'à un arbre $A = (V, E)$ muni d'une valuation L strictement positive de ses arêtes on peut associer une distance d_L définie comme l'application de V^2 vers $\mathbb{R}^+$ qui, à tout couple (x, y) de sommets de l'arbre, associe la somme des valeurs de toutes les arêtes situées sur le chemin unique reliant x à y. Ceci permet de donner une définition de la notion de représentation arborée et d'en déduire une catégorie de distances appelée distance additive d'arbre.

Définition 2.1.3 REPRÉSENTATION ARBORÉE.
Soient un ensemble X et d une distance sur X. On dira que (X, d) admet une ***représentation arborée*** *s'il existe un arbre valué $A = (V, E, L)$ qui vérifie les trois propriétés :*

1. $F \subseteq X \subseteq V$, *où F est l'ensemble des feuilles de A ;*
2. $\forall x \in (V - F) - X,\ N(x) \geq 3$, *où $V - F$ est l'ensemble des noeuds internes de A ;*
3. $\forall x, y \in X,\ d(x,y) = d_L(x,y)$.

La distance d sur X est alors appelée **distance additive d'arbre**, *ou* **distance arborée**.

Le lecteur intéressé par les différents aspects formels de cette définition pourra se reporter à [BG91].

Une condition nécessaire et suffisante pour qu'une distance soit une distance additive d'arbre est donnée par un théorème, dû à Zaretstskii et à Buneman (voir [Bun71], [BG91]).

Théorème 2.1.4 TOPOLOGIE ARBORÉE.
Soit X un ensemble et d une distance sur X. (X, d) admet une représentation arborée si et seulement si d respecte la **condition des quatre points** *: pour tout quadruplet $x, y, z, t \in X$, les deux plus grandes des trois sommes $d(x, y)+d(z, t)$, $d(x, z)+d(y, t)$ et $d(x, t)+d(y, z)$ sont égales.*
Nous dirons alors que (X, d) (ou, par abus d'écriture, X ou d) est muni d'une ***topologie arborée****.*

Cela permet une représentation sous forme d'arbre. En biologie, on essaie de classer les espèces, notamment dans le but d'établir des parentés évolutives. Pour cela, on établit des dissimilarités, principalement en mesurant l'éloignement biochimique. Ces parentés évolutives sont matérialisées par un arbre, appelé **arbre d'évolution**, **phylogénie**, ou encore **arbre phylogénétique**. L'établissement d'un arbre d'évolution peut répondre à des enjeux médicaux, pharmaceutiques, sociaux et économiques majeurs (voir, par exemple, [Bit00]).

Il existe plusieurs méthodes pour construire une phylogénie à partir d'une distance additive d'arbre (voir [BG91], [Bun71], [Hei89]), mais aussi à partir de mesures expérimentales (voir par exemple [BG00], [Bun71], [KHM97], [RG02]) par ajustement à une distance additive d'arbre.

Exemple 2.1.5
Afin de faciliter la compréhension du lecteur non spécialiste, nous allons présenter un exemple classique de dissimilarités établies pour 9 espèces de grenouilles. Les données ont été téléchargées sur la page web d'O. Gascuel : "*Logiciels et données disponibles*", URL : "http ://www.lirmm.fr/ gascuel/MAAS/#data" (lien : "QR2"), (consultée en juin 2002). La matrice de dissimilarité M est la suivante.

M	Aur.	Boy.	Casc.	Musc.	Temp.	Pret.	Cates.	Pip.
Aurora								
Boylii	10							
Cascadae	13	7						
Muscosa	12	7	7					
Temporaria	57	50	40	45				
Pretiosa	22	9	11	15	48			
Catesbeiana	86	65	54	48	85	54		
Pipiens	89	67	66	49	83	55	54	
Tarahumarae	97	72	79	67	107	60	59	48

Matrice de dissimilarité M

Un arbre phylogénétique reconstruit à partir de ces données est présenté dans la figure 2.1.
◇

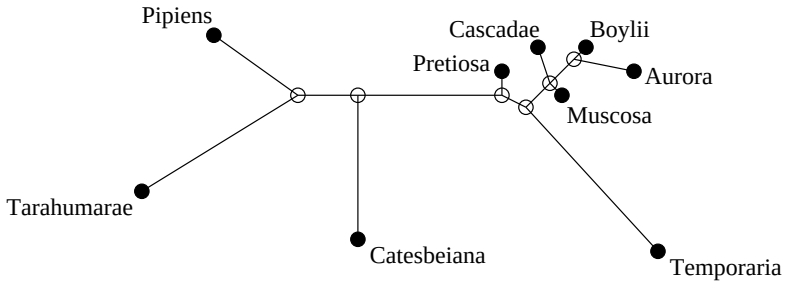


FIG. 2.1 – Un arbre phylogénétique pour les 9 grenouilles de l'exemple 2.1.5

Nous avons construit cet arbre par la méthode `NJ` *à l'aide du logiciel T-Rex, que nous avons téléchargé sur le site en ligne de P. Casgrain (voir [T-Rex]).*

2.1.2 Matrice ordinale associée à une dissimilarité

A chaque dissimilarité, on peut associer une matrice dite **ordinale** selon le procédé suivant. Chaque valeur de la dissimilarité δ sur l'ensemble X permet de définir un **seuil** : on classe par ordre croissant l'ensemble des valeurs de la dissimilarité, ce qui permet de constituer un ordre total dont les éléments, appelés seuils, sont numérotés de 0 (pour $\delta(x,x) = 0$) à k. Mathématiquement parlant, si M est la matrice de δ, l'ensemble des cases de M (et donc X^2) se structure en un préordre, appelé **préordonnance** ; le passage à l'ordre quotient canonique aboutit, sur une partition de X^2, à un ordre total, appelé **ordonnance**, dont les $k+1$ éléments sont numérotés de 0 à k.

Nous noterons σ l'application de $[0..k]$ vers $\mathbb{R}^+$ qui à chaque seuil i associe la i-ième valeur de la dissimilarité.

On définit alors une nouvelle matrice W associée à δ, appelée **matrice ordinale**, en remplaçant dans chaque case de M la valeur $\delta(x,y)$ par le seuil i correspondant tel que $\sigma(i) = \delta(x,y)$. La valeur $\sigma(i)$ associée au seuil i sera notée σ_i. Dans une matrice M de dissimilarité ou de seuils, nous présenterons le premier indice x d'une case $M[x,y]$ sur les colonnes et le deuxième indice y sur les lignes.

Nous dirons que δ est une **instanciation** de W. Plus généralement, nous dirons qu'une matrice symétrique à valeurs entières positives est **instanciable** quand il existe une dissimilarité dont elle est la matrice ordinale.

Exemple 2.1.6 Reprenons l'exemple 2.1.5.
Dans la matrice de dissimilarité M, les valeurs non nulles se répartissent en 29 seuils de 7 ($\sigma_1 = 7$) à 107 ($\sigma_{29} = 107$). Il y a trois paires correspondant au premier seuil : {Boylii, Cascadae}, {Boylii, Muscosa} et {Cascadae, Muscosa}. Le dernier seuil ne correspond qu'à une paire : {Temporaria, Tarahumarae}. La matrice ordinale correspondante W est la suivante.

W	Aur.	Boy.	Casc.	Musc.	Temp.	Pret.	Cates.	Pip.
Aurora								
Boylii	3							
Cascadae	6	1						
Muscosa	5	1	1					
Temporaria	16	13	9	10				
Pretiosa	8	2	4	7	11			
Catesbeiana	26	19	14	11	25	14		
Pipiens	27	21	20	12	24	15	14	
Tarahumarae	28	22	23	21	29	18	17	11

Matrice ordinale

◇

Il ne faut pas perdre de vue que le passage à la matrice ordinale provoque une perte d'information. En effet, si une matrice de dissimilarité correspond à une unique matrice ordinale, par contre, à une matrice ordinale peuvent correspondre de nombreuses matrices de dissimilarité, même si on peut conjecturer que les matrices de cette famille possèdent des propriétés communes. Les matrices ordinales peuvent cependant fournir des informations structurelles. Ces matrices, et les distances associées, sont étudiées dans plusieurs travaux (voir [BG91], [Gue98], [HNW99], [KHM97]).

Notre but est d'utiliser les matrices ordinales pour les traduire en terme de familles de graphes, ce qui fera l'objet de la sous-section suivante. Auparavant, nous allons donner quelques précisions sur le problème de l'instanciation d'une matrice ordinale.

Les propriétés ordinales d'une distance arborée ont été plus particulièrement étudiées par A. Guénoche. Il a notamment traité la question de l'instanciation d'une matrice ordinale en distance additive d'arbre, en relation avec la notion de *distance d'ordres*, et a proposé une "condition *ordinale* des quatre points" dans le but de caractériser les matrices ordinales instanciables en distance additive d'arbre (voir [Gue98]). Nous allons ici, sans passer par la notion de distance d'ordre, montrer qu'une matrice ordinale est toujours instanciable en distance, puis établir une condition d'instanciation d'une matrice ordinale en distance additive d'arbre.

Donnons tout d'abord une définition formelle à la notion de matrice ordinale :

Définition 2.1.7 MATRICE ORDINALE, MATRICE INTÈGRE
On appellera **matrice ordinale** *toute matrice sur un ensemble X qui est symétrique et dont l'ensemble des valeurs forme un intervalle $[0..k]$ d'entiers (tous les entiers compris entre 0 et k sont présents dans la matrice, et seulement ceux-là).*
Une matrice ordinale W sur un ensemble X sera dite **intègre** *quand $\forall x, y \in X$, $W[x, y] = 0 \iff x = y$.*

Tout d'abord, il est évident que toute matrice ordinale W, définie sur un ensemble X, est instanciable : les valeurs de W constituent une dissimilarité sur X (définition 2.1.1). Par construction de la matrice ordinale associée à une dissimilarité, il est aussi évident qu'une matrice ordinale W est instanciable en indice de distance (définition 2.1.2) si et seulement si elle est intègre. Il existe alors une instanciation de W en distance :

Propriété 2.1.8

- *Si une matrice ordinale W sur un ensemble X est instanciable en indice de distance alors elle est instanciable en une distance d sur X.*
- *De plus, toute application Δ de X^2 vers IR, définie pour une constante donnée $C \in IR$ par $\Delta(x,x) = 0$ et, pour $x \neq y$, par $\Delta(x,y) = d(x,y) + C$, est une distance ayant W pour matrice ordinale.*

Preuve : Soit W une matrice ordinale sur un ensemble X, avec $k+1$ seuils $0, ..., k$.

- On sait déjà qu'il existe un indice de distance δ sur X défini par $\delta(x,y) = W[x,y]$. Définissons alors l'application σ de $[0..k]$ vers $\mathbb{N}$ telle que $\sigma(0) = 0$ et, pour $i \in [1..k]$, $\sigma(i) = k+i$. Puisque $\sigma(i) = 0 \iff i = 0$, on a donc $\forall x, y \in X$, $\sigma(W[x,y]) = 0 \iff x = y$. Ainsi, l'application d de X^2 vers $\mathbb{N}$ définie par $\forall x, y \in X$, $d(x,y) = \sigma(W[x,y])$ est un indice d'écart et, pour $x \neq y$, on a $k \leq d(x,y) \leq 2k$. Par conséquent, pour $x, y, z \in X$, on a $d(x,y)+d(y,z) \geq 2k \geq d(x,z)$, ce qui correspond à l'inégalité triangulaire. En conclusion, d est une distance sur X ayant W pour matrice ordinale. W est donc instanciable en distance.
- On peut reprendre cette démonstration en changeant la constante permettant de passer de l'indice de distance à la distance : $\sigma'(i \neq 0) = C+k+i$ permet de définir une nouvelle distance Δ telle que $\forall x \in X$, $\Delta(x,x) = 0$ et $\forall x, y \in X$, $x \neq y$, $\Delta(x,y) = d(x,y) + C$.

◇

Même si une distance n'est pas arborée, sa matrice ordinale peut être instanciable en distance arborée, comme le montre l'exemple 2.1.9. Il existe aussi des matrices ordinales qui ne sont pas instanciables en distance arborée, c'est ce que montre le contre-exemple 2.1.10.

Exemple 2.1.9

La distance définie sur $X = \{a, b, c, d\}$ par la matrice M_1 ci-dessous n'est pas arborée : $ab+cd>ac+bd>ad+bc$ (10>9>9), la condition des quatre points n'est donc pas respectée.

M_1	b	c	d
a	8	6	5
b		4	3
c			2

Matrice M_1 de distance

W_1	b	c	d
a	6	5	4
b		3	2
c			1

Matrice ordinale W_1 associée

La matrice ordinale W_1 est pourtant instanciable en distance arborée :

W_1	b	c	d
a	6	5	4
b		3	2
c			1

La matrice ordinale W_1

M_1'	b	c	d
a	18	17	15
b		14	13
c			12

Une instanciation M_1' en distance arborée

Dans M_1', on a $ab + cd = ac + bd \geq ad + bc$ ($30 \geq 29$).

◇

Contre-exemple 2.1.10

La matrice ordinale ci-dessous n'est pas instanciable en distance arborée.

En effet, l'inégalité stricte $\sigma_1+\sigma_4 < \sigma_2+\sigma_5 < \sigma_3+\sigma_6$ est vraie quelles que soient les valeurs σ_i. La condition des 4 points ne peut donc pas être respectée, quelle que soit l'instanciation de W.

W_2	b	c	d
a	1	6	2
b		5	3
c			4

◇

On sait qu'un indice de distance qui respecte la condition des quatre points respecte aussi l'inégalité triangulaire et constitue donc une distance arborée. Cette implication peut, dans une certaine mesure, être transposée en termes de matrice ordinale.

Propriété 2.1.11
Si une matrice ordinale intègre W vérifie la propriété suivante : $\forall x, y, z, t \in X$ les deux plus grandes des trois sommes $W[x, y] + W[z, t]$, $W[x, z] + W[y, t]$ et $W[x, t] + W[y, z]$ sont égales alors elle est instanciable en une distance arborée.

Preuve : Soit, sur un ensemble X, une matrice ordinale W intègre dont l'ensemble de valeurs forme l'ensemble $[0..k]$ et qui respecte la propriété : $\forall x, y, z, t \in X$ les deux plus grandes des trois sommes $W[x, y] + W[z, t]$, $W[x, z] + W[y, t]$ et $W[x, t] + W[y, z]$ sont égales.
On sait, par la propriété 2.1.8, que W est la matrice ordinale de la distance d définie par $\forall x \in X$, $d(x, x) = 0$ et $\forall x, y \in X$, $x \neq y$, $d(x, y) = W[x, y] + k$. Supposons que pour quatre éléments x, y, z et t de X la propriété sur W s'exprime par $W[x, y] + W[z, t] = W[x, z] + W[y, t] \geq W[x, t] + W[y, z]$.

- Si x, y, z et t sont deux à deux distincts, on a $W[x, y] \neq 0$ et donc $d(x, y) = W[x, y]+k$, de même pour les cinq autres valeurs de W intervenant dans les sommes. On a donc $d(x, y) + d(z, t) = W[x, y] + W[z, t] + 2k = W[x, z] + W[y, t] + 2k = d(x, z) + d(y, t)$, mais aussi $W[x, z] + W[y, t] + 2k \geq W[x, t] + W[y, z] + 2k$ et donc $d(x, z) + d(y, t) \geq d(x, t) + d(y, z)$.
- Si $x = y$ et $z = t$ alors $W[x, y] + W[z, t] = 0$ donc $x = y = z = t$ et $d(x, y) + d(z, t) = d(x, z) + d(y, t) = d(x, t) + d(y, z) = 0$.
- Si on a seulement $x = y$, on a alors $W[x, y] = 0$ et la propriété sur W devient $W[z, t] = W[x, z] + W[y, t]$ avec $W[x, z] + W[y, t] = W[x, t] + W[y, z]$. On a donc $W[z, t] + k \leq W[x, z] + W[y, t] + 2k$ et donc $d(z, t) \leq d(x, z) + d(y, t)$. Puisque $x = y$, on en déduit $d(x, y) + d(z, t) \leq d(x, z) + d(y, t) = d(x, t) + d(y, z)$. Le raisonnement est semblable si on a seulement $z = t$, $x = z$ ou bien $y = t$.
- Si on a seulement $x = t$, on aura $d(x, t) + d(y, z) = d(y, z) = W[y, z] + k$ et donc $d(x, t) + d(y, z) \leq d(x, z) + d(y, t)$; l'égalité des deux plus grandes sommes $d(x, y) + d(z, t)$ et $d(x, z) + d(y, t)$ est démontrée comme dans le premier cas. Le raisonnement est semblable si on a seulement $y = z$.
- Si on a plusieurs égalités :
  - $x = y = z = t$: on a alors $d(x, y) + d(z, t) = d(x, z) + d(y, t) = d(x, t) + d(y, z) = 0$.
  - $x = y$ et $z = t$: la propriété devient $0 = W[x, z] + W[y, t] \geq W[x, t] + W[y, z]$, on a donc $W[x, z] = 0$ et $W[y, t] = 0$ donc $x = y = z = t$, on se ramène au cas précédent. Le raisonnement est semblable si $x = z$ et $y = t$.

- $x = t$ et $y = z$: $d(x,t) + d(y,z) = 0$ est la plus petite des trois sommes, les deux autres étant égales, ce qu'on démontre sur le modèle du premier cas.
- $x = y = z$: on a $d(x,t) + d(y,z) = d(x,t)$, $d(x,y) + d(z,t) = d(z,t) = d(x,t)$ et $d(x,z) + d(y,t) = d(y,t) = d(x,t)$. Les trois sommes sont donc égales. Le raisonnement est semblable pour $x = z = t$, $x = y = t$ et $y = z = t$.

On voit donc que la distance d respecte la condition des quatre points et constitue une distance additive d'arbre.
$\diamond$

2.1.3 Famille de graphes emboîtés associée à une matrice ordinale

Comme une matrice de dissimilarité – ainsi que la matrice ordinale associée – est symétrique, il est naturel de l'associer à des graphes.

Définition 2.1.12 GRAPHES EMBOÎTÉS ASSOCIÉS À UNE DISSIMILARITÉ.
Soit X un ensemble, δ une dissimilarité sur X et W la matrice ordinale correspondante définissant les seuils $0, ..., k$. On définit une famille de graphes $G_0 \subset G_1 \subset ... \subset G_k$ emboîtés associée à W, où chaque graphe $G_i = (V, E_i)$ est défini comme suit :

1. *$V = X$;*
2. *pour $x, y \in X$, avec $x \neq y$, $xy \in E_i \Longleftrightarrow \delta(x,y) \leq \sigma_i$.*

*Nous appellerons cette famille : **famille de graphes emboîtés** (*threshold family of graphs*) définie par une matrice de dissimilarité, ou plus simplement, quand il n'y a pas d'ambiguïté, **famille emboîtée**.*

Remarque 2.1.13 Ces "*graphes seuils*", définis à partir d'une suite de seuils, ne doivent pas être confondus avec les *threshold graphs* tels qu'ils ont été définis dans le contexte de la programmation en nombres entiers. Les *threshold graphs* ont été caractérisés comme étant triangulés, co-triangulés et sans-P_4 (voir [Gol80] et [BLS99]) ce qui n'est absolument pas le cas généralement des graphes emboîtés de la définition 2.1.12.

La dissimilarité induit donc, sur $V \times V$, un préordre (correspondant au préordre ayant fourni les seuils) et donc une partition ordonnée des arêtes des graphes G_k ; chacune des classes F_i de cette partition étant définie et caractérisée par $F_i = E_i - E_{i-1} = \{xy \in V \times V \mid \delta(x,y) = \sigma_i\}$. Le graphe G_i est donc obtenu en ajoutant l'ensemble d'arêtes F_i au graphe G_{i-1}. L'ensemble des classes F_i est donc totalement ordonné par les seuils i.

Remarque 2.1.14
Les valeurs $W[x,x]$ d'une matrice ordinale W (et par conséquent les valeurs $\delta(x,x)$ de la dissimilarité δ dont elle est issue) n'interviennent pas dans la construction des graphes emboîtés qui ne contiennent pas de boucles.

Si une dissimilarité δ possède une valeur nulle $\delta(x,y) = 0$ pour $x \neq y$, on retrouvera cette valeur nulle $W[x,y] = 0$ dans la matrice ordinale W correspondante. Le premier graphe emboîté G_0 contiendra alors l'arête xy. Afin d'uniformiser les familles de graphes emboîtés, nous allons prendre la convention suivante :

- Le premier graphe G_0 d'une famille emboîtée est toujours un stable ;
- S'il existe dans la matrice ordinale W des valeurs $W[x, y] = 0$ pour $x \neq y$ alors le graphe G_1 contient les arêtes xy correspondantes, le graphe G_2 correspondant alors au premier seuil non nul de W.

Cette convention ajuste implicitement la matrice ordinale W à une matrice ordinale intègre.

Exemple 2.1.15
Reprenons l'exemple 2.1.5 sur les grenouilles.

Les graphes de la famille associée qui correspondent aux 5 premiers seuils non nuls sont donnés dans la figure 2.2.

Pour des raisons de lisibilité, nous avons remplacé, dans les graphes, les noms de grenouilles par des lettres selon la correspondance suivante :

Aurora	Boylii	Cascadae	Muscosa	Temporaria	Pretiosa	Catesbeiana	Pipiens	Tarahumarae
a	b	c	d	e	f	g	h	i

Seuil 1 : on ajoute les arêtes bc, bd et cd ;
seuil 2 : arête bf ;
Seuil 3 : arête ab ;
Seuil 4 : arête cf ;
Seuil 5 : arête ad ;
etc.
◇

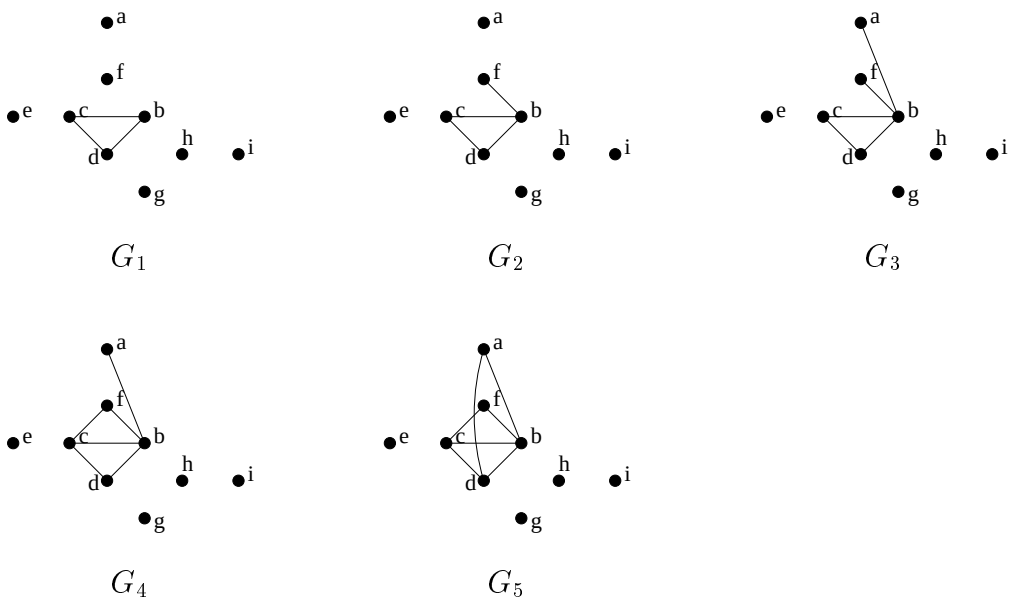


FIG. 2.2 – Les cinq premiers graphes de l'exemple 2.1.15.

2.2 Les distances triangulées

Il suffit qu'une distance soit arborée pour que tous les graphes de la famille associée soient triangulés :

Théorème 2.2.1
Soit δ une dissimilarité sur X et $(G_i)_{i\in[0..k]}$ la famille de graphes emboîtés associée. Si (X,δ) peut être représentée par une phylogénie alors chaque graphe G_i est triangulé.

Ce théorème est utilisé dans [HNW99] sans que la preuve y soit donnée. Nous en proposons une ici, qui est construite sur une caractérisation des graphes triangulés basée sur les arbres et découverte indépendemment par Buneman, Gàvril et Walter :

Caractérisation 2.2.2 *([Wal72], [Bun74], [Gav74]).*
Un graphe est triangulé si et seulement si il est le graphe d'intersection d'une famille de sous-arbres d'un arbre.

Preuve : (du théorème 2.2.1).
Soit X un ensemble d'objets, d une distance additive sur X et A la phylogénie associée. On peut facilement ajouter des sommets internes à A afin d'obtenir un nouvel arbre A' dans lequel il existe toujours un noeud à mi-distance de deux sommets donnés de X. Soit G_i le graphe associé au seuil i de d. Considérons alors la famille de sous-arbres T'_x de T' définie par : pour tout $x \in X$, T'_x est le sous-arbre de T' contenant tous les noeuds situés à une distance d'au plus $\frac{i}{2}$ de x. On a ainsi une famille de sous-arbres respectant la caractérisation 2.2.2 et G_i est triangulé.
$\diamond$

La réciproque du théorème 2.2.1 est fausse. Les distances additives ne sont pas les seules distances à définir une famille de graphes triangulés. Le théorème 2.2.1 peut être transposé en termes de matrice ordinale : si une matrice ordinale est instanciable en distance arborée alors tous les graphes de la famille associée sont triangulés. La réciproque reste fausse : une matrice ordinale peut être associée à une famille de graphes triangulés même si elle n'est pas instanciable en distance arborée, comme le montre le contre-exemple 2.2.3.

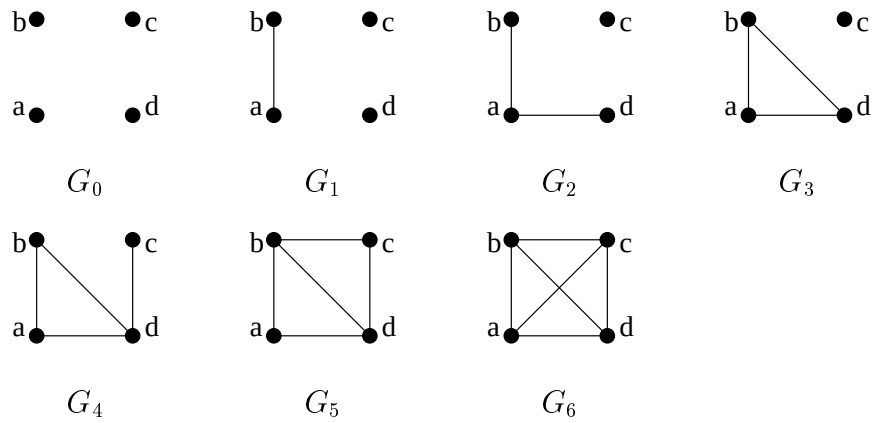


FIG. 2.3 – Famille de graphes associée à la dissimilarité du contre-exemple 2.2.3.

Contre-exemple 2.2.3
La matrice ordinale W_2 du contre-exemple 2.1.10 n'est pas instanciable en distance arborée.

W_2	b	c	d
a	1	6	2
b		5	3
c			4

Matrice ordinale W_2

Pourtant elle est associée à une famille de graphes triangulés comme le montre la figure 2.3.
◇

Nous pouvons donc définir une nouvelle classe de distances : la classe des **distances triangulées**, plus générale que la classe des distances additives d'arbres :

Définition 2.2.4 Distance triangulée.
*Nous appellerons **distance triangulée** une distance pour laquelle tous les graphes emboîtés de la famille associée sont triangulés.*

Il peut être intéressant de se ramener à une famille de graphes triangulés, car ces graphes possèdent de propriétés structurelles puissantes, en particulier :
- Les séparateurs minimaux sont complets et il y en a peu, ce qui permet d'utiliser la décomposition par séparateurs minimaux complets en choisissant les séparateurs les plus intéressants.
- Les cliques maximales possèdent une structure particulière.

Nous proposerons, dans le chapitre suivant, un procédé d'ajustement d'une dissimilarité à une distance triangulée.

Chapitre 3

Ajustements de données

Après avoir posé dans le chapitre précédent les fondations théoriques qui sont le point de départ de notre travail, nous allons montrer comment nous pouvons utiliser les outils de graphes dont nous disposons pour ajuster les données destinées à la construction de phylogénies.

L'examen de données expérimentales montre que non seulement les dissimilarités mesurées ne sont très généralement pas arborées, mais que, de plus, les graphes emboîtés sont loin d'être tous triangulés. Nous allons donc proposer un ajustement de données par triangulation de graphes.

L'approche classique pour trianguler un graphe, comme nous l'avons dit au chapitre 1, consiste à ajouter des arêtes cordant les cycles trop longs. Pour une famille de graphes associée à une dissimilarité δ, l'ajout d'une arête xy à un graphe revient à diminuer la valeur de dissimilarité $\delta(x, y)$ correspondante. Cependant, les biologistes estiment que, dans le contexte de la construction d'une phylogénie à partir de séquences d'ADN, les seuils obtenus par les expérimentations tendent plutôt à être sous-estimés, puisque le nombre de mutations, représenté par la distance entre taxa, est en fait plus élevé que ce qui est observé expérimentalement (voir [KHM97]).

Il est donc préférable – même lorsque le nombre de corrections nécessaires pour obtenir une famille de graphes triangulés est a priori faible – d'éviter de diminuer des seuils comme le propose [HNW99] en utilisant un procédé classique de triangulation, mais d'essayer au contraire d'augmenter ces valeurs pour compenser la sous-évaluation intrinsèque.

Remarquons que [HNW99] utilise un procédé qui fournit une triangulation "brutale", qui ajoute beaucoup plus d'arêtes que nécessaire à l'obtention d'une triangulation minimale. Un procédé tel que celui décrit dans [Ber99] (algorithme `LB-TRIANG`, voir page 46) serait beaucoup plus fin puisque, même lorsque l'ordre sur les sommets est imposé, il fournit une triangulation qui est minimale. On pourrait d'ailleurs, pour d'autres problématiques liées à des matrices ordinales, proposer différentes approches de triangulation, suivant le problème à résoudre.

Dans le cadre du problème posé ci-dessus, nous avons examiné la question de la triangulation d'un graphe par **retraits** d'arêtes plutôt que par **ajouts**. Ce procédé correspond à un concept de graphe nouveau que nous introduisons sous le nom de **sous-triangulation maximale**.

En conséquence, nous proposons un algorithme qui ajuste une dissimilarité par augmentation des seuils lorsqu'une une anomalie est détectée. Notre procédé construit une suite de sous-triangulations maximales des graphes correspondants définis par la matrice de départ ; plus généralement, on peut utiliser le procédé que nous décrivons pour calculer une sous-triangulation maximale d'un graphe quelconque.

3.1 Ajuster les données en triangulant

3.1.1 Un schéma de composition de graphes par arêtes

Afin de calculer une famille de graphes triangulés dont chaque graphe H_i soit un sous-graphe du graphe originel correspondant G_i, nous allons construire la clique H_k depuis le stable H_0 en ajoutant à chaque étape un ensemble d'arêtes, minimal au sens de l'inclusion, qui préserve la propriété d'être triangulé. Nous allons montrer que ceci peut être réalisé de façon équivalente en ajoutant les arêtes une par une tout en maintenant un graphe triangulé.

Pour cela, nous définissons pour les graphes triangulés un schéma de composition par arêtes.

Un schéma de composition par arêtes a été défini par Hayward ([Hay96]) pour la sur-classe des graphes triangulés que constituent les graphes faiblement triangulés. Nous modifions ce schéma d'ajout d'arêtes en utilisant un autre concept introduit pour les graphes faiblement triangulés : la notion de 2-paire, définie par Hayward, Hoàng et Maffray ([HHM89]) pour caractériser les graphes faiblement triangulés.

Définition 3.1.1 *([HHM89]).*
Dans un graphe, une paire $\{x, y\}$ de sommets non adjacents est appelée 2-paire quand tout chemin sans corde reliant x et y est de longueur exactement 2.

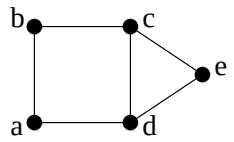


FIG. 3.1 – Le graphe de l'exemple 3.1.2.

Exemple 3.1.2
Dans le graphe non triangulé de la figure 3.1, $\{a, c\}$ est une 2-paire car les deux chemins

sans corde abc et adc qui relient a et c sont de longueur 2. Par contre, $\{a, e\}$ n'est pas une 2-paire : $abce$ est un chemin sans corde de longueur supérieure à 2. $\{c, d\}$ n'est pas une 2-paire car c et d sont adjacents.
◇

Nous proposons le schéma de composition suivant, qui caractérise les graphes triangulés.

Schéma de composition 3.1.3
Un graphe à n sommets est triangulé si et seulement si il peut être construit en commençant par un stable à n sommets et en ajoutant à chaque étape une arête entre deux sommets qui forment une 2-paire.

Ce schéma de composition respecte l'invariant suivant :

Invariant 3.1.4
A chaque étape du schéma de composition 3.1.3, le nouveau graphe obtenu est triangulé.

Cet invariant repose sur le théorème suivant :

Théorème 3.1.5
Soient G_1 un graphe triangulé et $\{a, b\}$ une paire de sommets non-adjacents de G_1, soit G_2 le graphe obtenu à partir de G_1 en lui ajoutant l'arête ab. G_2 est triangulé si et seulement si $\{a, b\}$ est une 2-paire de G_1.

Preuve : Soient G_1 un graphe triangulé et $\{a, b\}$ une paire de sommets non-adjacents de G_1, soit G_2 le graphe obtenu à partir de G_1 en lui ajoutant l'arête ab.
Considérons un quelconque plus long chemin sans corde $\mu = ax_1...x_kb$ de a vers b dans G_1. Dans G_2, μa sera un cycle sans corde ayant plus de 3 sommets (cycle sans corde qui empêchera G_2 d'être triangulé) si et seulement si μ est de longueur strictement supérieure à 2, c'est-à-dire si et seulement si $\{a, b\}$ n'est pas une 2-paire de G_1.
◇

Pour assurer le passage d'un graphe triangulé H_i au graphe triangulé suivant H_{i+1} dans la famille de triangulés que nous cherchons à construire, nous avons besoin de la propriété suivante.

Propriété 3.1.6
Soit G_1 et G_2 deux graphes triangulés tels que $G_1 \subset G_2$. Alors G_2 peut être obtenu à partir de G_1 par ajouts successifs d'une arête entre deux sommets formant une 2-paire du graphe transitoire.

Remarquons que cette propriété est loin d'être trivialement extensible à toute classe de graphes : elle n'est, par exemple, pas valide pour les graphes sans trous, comme le montre l'exemple 3.1.7 (pour lequel nous remercions G. Simonet). On ne sait d'ailleurs pas si elle est valide pour les graphes faiblement triangulés, ce qui constitue une des questions ouvertes qu'il nous reste sur cette classe.

Contre-exemple 3.1.7
Les graphes G_1 et G_2 de la figure 3.2 sont sans trou. Pour passer de G_1 à G_2, il faut ajouter les arêtes ac et df. Cependant, les graphes intermédiaires $G_1 + \{ac\}$ et $G_1 + \{df\}$ ont un trou, respectivement $facdef$ et $abcdfa$.
◇

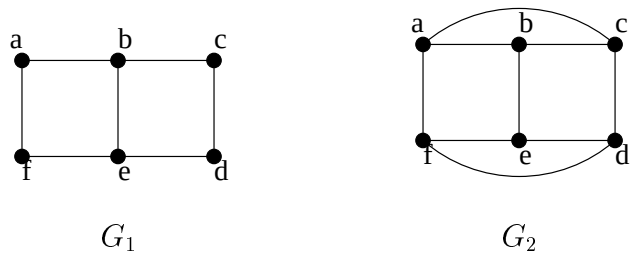


FIG. 3.2 – Les graphes de l'exemple 3.1.7.

Pour prouver la propriété 3.1.6, nous utilisons le lemme suivant, qui a été prouvé par Rose, Tarjan et Lueker pour définir des outils fondamentaux pour la triangulation minimale.

Lemme 3.1.8 *([RTL76]).*
Soient $G_1 = (V, E_1)$ et $G_2 = (V, E_2)$ deux graphes triangulés tels que $G_1 \subset G_2$. Alors il existe dans $E_2 - E_1$ une arête f telle que le graphe $G' = (V, E_2 - \{f\})$ soit triangulé.

Preuve : (de la propriété 3.1.6).
Soit G_1 et G_2 deux graphes triangulés tels que $G_1 \subseteq G_2$. On sait, par le lemme 3.1.8 qu'il existe dans $E_2 - E_1$ une arête ab telle que $(V, E_2 - \{ab\})$ est triangulé. Par le théorème 3.1.5, $\{a, b\}$ est une 2-paire de $G_2 - \{ab\}$. Si nous répétons ceci jusqu'à obtenir le graphe G_1, nous aurons construit – à l'envers – un ordre d'addition d'arête 2-paire qui permet de construire G_2 à partir de G_1.
◇

Le schéma de composition 3.1.3 découle trivialement de la propriété 3.1.6.

Exemple 3.1.9
Dans la figure 3.3, on peut passer d'un graphe H_1 au suivant par ajout d'une 2-paire.
$\{c, d\}$ est une 2-paire de H_1 car l'unique chemin ced entre c et d est de longueur 2 ; On passe de H_1 à H_2 en ajoutant cd.
$\{b, d\}$ est une 2-paire de H_2, on passe de H_2 à H_3 en ajoutant l'arête bd.
Enfin, $\{b, c\}$ est une 2-paire de H_3, on passe de H_3 à H_3 en ajoutant l'arête bc.
◇

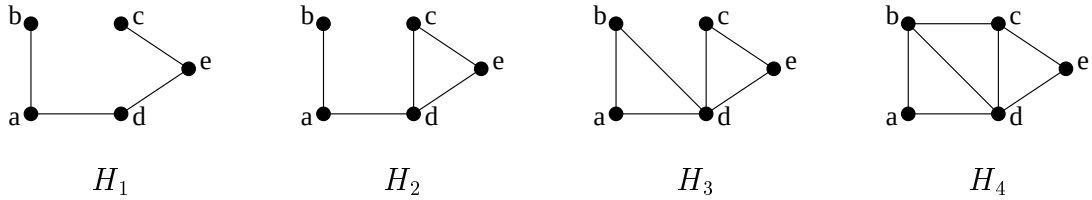


FIG. 3.3 – Les graphes de l'exemple 3.1.9.

3.1.2 Un nouveau concept de graphes : la sous-triangulation maximale

Ce qui précède nous amène au concept nouveau de sous-triangulation maximale :

Définition 3.1.10 SOUS-TRIANGULATION MAXIMALE
Soient $G = (V, E)$ un graphe non triangulé et $H = (V, F)$ un graphe triangulé tel que $F \subset E$. Nous dirons que H est une ***sous-triangulation maximale*** *de G quand, pour toute partie propre F' de $E-F$, le graphe $(V, F+F')$ n'est pas triangulé.*

Le calcul d'une sous-triangulation maximale d'un graphe $G = (V, E)$ peut être réalisé en temps polynomial en utilisant le schéma de composition 3.1.3 :

Procédé algorithmique 3.1.11

Donnée : un graphe non triangulé $G = (V, E)$.
Résultat : une sous-triangulation maximale $H = (V, F)$ de G.
Début
$F \leftarrow \emptyset$;
Tant que il y a une paire $\{a, b\}$ de $E-F$ qui est une 2-paire du graphe (V, F) **faire :**
  $F \leftarrow F+\{ab\}$;
Fin.

Preuve de la correction du procédé algorithmique 3.1.11 :
Il est clair qu'à la fin de l'algorithme, le graphe $H = (V, F) \subset G$ est triangulé puisqu'il est obtenu par ajouts successifs de 2-paires (théorème 3.1.5). Supposons que ce graphe H ne soit pas une sous-triangulation maximale du graphe non triangulé G ; cela signifie qu'il existe un graphe triangulé H', avec $H \subset H' \subset G$. La propriété 3.1.6 peut être appliquée aux graphes emboîtés H' et $H \subset H'$: H' peut être obtenu à partir de H par un ajout successif de 2-paires. Ceci est contradictoire : l'algorithme ne pouvait donc pas s'arrêter sur le graphe H, mais continuait, jusqu'à obtenir une sous-triangulation maximale de G.
◇

3.1.3 Un algorithme de sous-triangulation de la famille de graphe définie par une matrice ordinale

Nous proposons maintenant un algorithme basé sur le schéma de composition 3.1.3 qui prend en entrée une matrice de dissimilarité M et qui retourne une matrice de dissimilarité M' définissant une famille de graphes triangulés obtenue par augmentation des seuils modifiés de M.

Notre algorithme est initialisé avec un stable H_O. A l'étape i l'algorithme choisit, parmi un ensemble de paires candidates, une 2-paire et ajoute l'arête correspondante, en privilégiant les 2-paires les plus anciennes (avec le seuil le plus petit) de façon à modifier le moins possible la matrice initiale. L'ensemble des 2-paires est alors mis à jour, et ce procédé est répété jusqu'à ce qu'il n'y ait plus de 2-paires. On passe alors à l'étape $i+1$, ce qui provoque un apport de paires candidates parmi lesquelles on a de nouveau une chance de trouver une 2-paire. Remarquons que l'ajout d'une 2-paire peut provoquer l'apparition de 2-paires parmi les paires candidates précédemment écartées.

Afin de donner priorité parmi les paires candidates à celles qui ont le seuil le plus bas, nous utilisons une file. A chaque étape i, les nouvelles paires candidates sont ajoutées à la file, puis l'algorithme choisit la première paire de la file qui soit une 2-paire du graphe courant et l'ajoute au graphe en cours de construction à partir du graphe courant.

D'après la propriété 3.1.6, à la fin de l'algorithme, la file est vide et un seuil a été attribué à chaque paire de taxa dans la matrice M' obtenue.

Algorithme `ADD-SUB-TRI` :

Donnée : une matrice M d'une dissimilarité sur n objets.
Résultat : une matrice de dissimilarité M' telle que tous les graphes de la famille associée soient triangulés.
Initialisation :
Construire la matrice ordinale associée à M, définissant les seuils $0..k$;
// La fonction σ associe à chaque seuil i sa valeur de dissimilarité σ_i.
$H_0 \leftarrow$ stable à n sommets ;
Créer une file vide FILE ;
Début
Pour i **de** 1 **à** $k-1$ **faire :**
$H_i \leftarrow H_{i-1}$;
Calculer l'ensemble P_i des paires $\{a, b\}$ telles que $M[a, b] = \sigma_i$;
Ajouter P_i à FILE ;
Répéter :
Retirer de FILE la première paire $\{a, b\}$ qui est une 2-paire ;
$M'[a, b] \leftarrow \sigma_i$; Ajouter l'arête ab au graphe H_i ;
jusqu'à : FILE ne contient plus de 2-paire de H_i ;
$H_k \leftarrow H_{k-1}$;
Pour toute paire $\{a, b\}$ restant dans FILE **faire :**
$M'[a, b] \leftarrow \sigma_k$; Ajouter l'arête ab à H_k ; *//H_k forme une clique.*
Fin.

Remarquons que l'algorithme `ADD-SUB-TRI` n'est pas déterministe ; en effet, à chaque étape, le choix d'une 2-paire candidate n'est pas unique, puisqu'il dépend de l'ordre dans lequel on a placé dans la file les paires, a priori équivalentes, fournies par le seuil considéré. Cependant, en vertu du procédé algorithmique 3.1.11, l'algorithme calcule dans tous les cas une matrice de dissimilarité telle que chaque graphe de la famille associée constitue une sous-triangulation maximale du graphe correspondant dans la matrice de dissimilarité initiale.

Complexité de l'algorithme :

Afin d'obtenir une meilleure complexité, la file contenant les arêtes candidates peut être associée à une structure de données gérant les 2-paires. Dans [SS95], Spinrad et Sritharan ont proposé un algorithme d'ajout itératif de 2-paires à un graphe ; ils utilisent une structure de données qui maintient la "structure des 2-paires" du graphe, ce qui coûte $O(n^2)$ de mise à jour à chaque ajout d'arête. Comme il y a $O(n^2)$ arêtes à gérer, en se basant sur cette structure, la complexité globale de notre algorithme est $O(n^4)$.

Remarquons que [SS95] traite le problème du maintien d'une structure des 2-paires dans un graphe quelconque, alors que nous travaillons seulement sur des graphes triangulés. Dans un graphe triangulé, il est facile de montrer que $\{a, b\}$ est une 2-paire si et seulement si a et b sont des points de confluence pour deux composantes pleines différentes définies par un séparateur minimal du graphe (un point de confluence d'une composante connexe C est définie dans [BBH00] comme un sommet $x \in C$ tel que $N(C) \subset N(x)$). Par conséquent, l'ensemble des 2-paires d'un graphe triangulé peut être maintenu plus facilement que dans un graphe quelconque ; il est raisonnable de penser que l'ensemble des 2-paires d'un graphe triangulé peut être mis à jour en $O(m)$ (où m est le nombre d'arêtes du graphe) ; ceci n'améliore cependant pas notre complexité globale dans le pire cas, puisque m atteint $O(n^2)$. Notons aussi que dans les graphes triangulés il peut y avoir des 2-paires qui ne sont pas perturbées par un ajout d'arête, ce qui fait qu'il n'est pas forcément nécessaire de mettre à jour la structure des 2-paires à chaque étape.

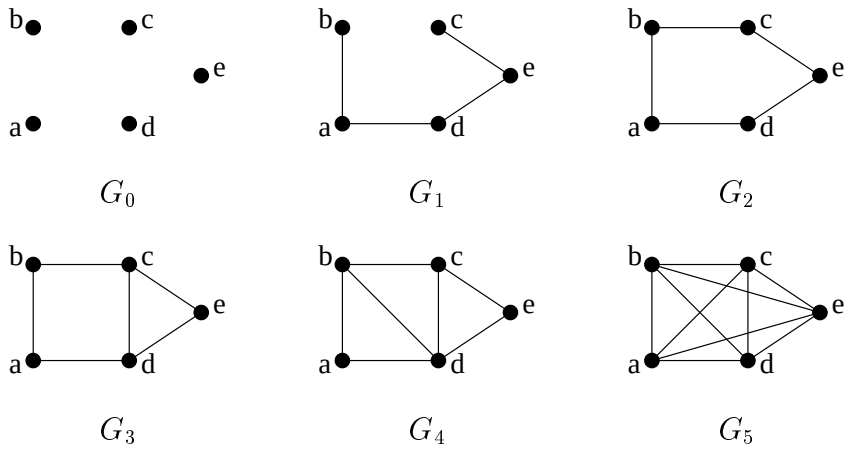


FIG. 3.4 – La famille de graphes initiale de l'exemple 3.1.12.

Exemple 3.1.12
Exécutons `ADD-SUB-TRI` sur la matrice ordinale d'une dissimilarité non triangulée sur cinq objets $\{a, b, c, d, e\}$ dont la famille de six graphes associée est donnée dans la figure 3.4.

Dans cette famille, G_2 et G_3 ne sont pas triangulés.
Initialisation : au seuil 0, la file est vide, H_0 est un stable.
Etape 1 : Au seuil 1, les paires candidates $\{a, b\}$, $\{a, d\}$, $\{c, e\}$ et $\{d, e\}$ sont placées dans la file. $\{a, b\}$ est une 2-paire du graphe H_0 et sort de la file pour être ajoutée au graphe ; $\{a, d\}$ est une 2-paire de $H_0+\{ab\}$ et sort de la file pour être ajoutée au graphe ; on continue de même avec $\{c, e\}$ et $\{d, e\}$. On obtient $H_1 = G_1$. La file est à nouveau vide.
Etape 2 : Au seuil 2, la paire candidate $\{b, c\}$ est placée dans la file. Ce n'est pas une 2-paire de H_1. Il n'y a pas d'autre arête candidate à ce seuil. $H_2 = H_1$.
Etape 3 : La paire $\{c, d\}$ est placée dans la file. En début de file, $\{b, c\}$ n'est pas une 2-paire de H_2, au contraire de son successeur $\{c, d\}$ qui sort de la file. On obtient $H_3 = H_2+\{cd\}$. $\{b, c\}$ n'est pas non plus une 2-paire de H_3, elle reste donc dans la file.
Etape 4 : La paire $\{b, d\}$ est placée dans la file. $\{b, c\}$ n'est pas une 2-paire de H_3, au contraire de son nouveau successeur $\{b, d\}$ qui sort de la file. On obtient $H_4 = H_3+\{bd\}$. $\{b, c\}$ devient enfin une 2-paire de H_4, on obtient $H_4 = H_3+\{bd, bc\}$. La file est vide.
Etape 5 : Enfin, au seuil 5, les paires candidates $\{b, e\}$ et $\{a, e\}$ sont placées dans la file. Elles en ressortent successivement pour former la clique $H_5 = G_5$.
Il n'y a plus d'arête candidate, l'algorithme prend fin.
L'algorithme a donc déplacé une arête, bc du seuil 1 au seuil 4. La nouvelle famille de graphes est donnée dans la figure 3.5. Notons qu'on retrouve les graphes de l'exemple 3.1.9.
◇

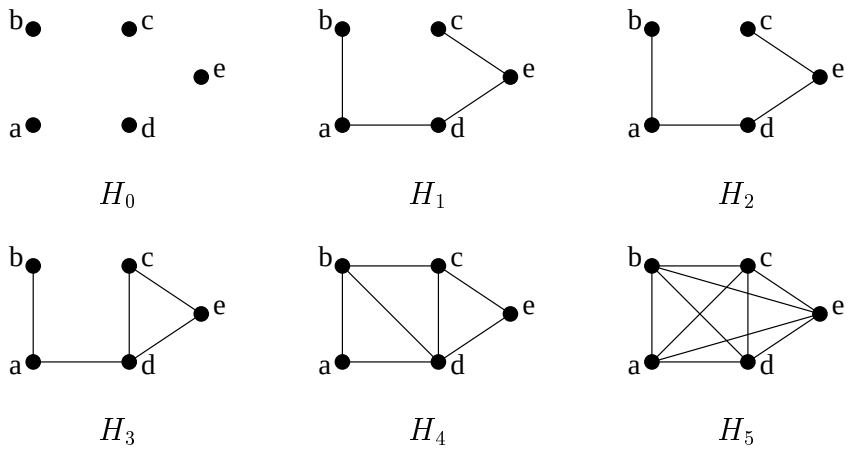


FIG. 3.5 – La famille de graphes obtenue dans l'exemple 3.1.12 par une exécution de l'algorithme `ADD-SUB-TREE` sur la famille de graphes de la figure 3.4.

Nous avons testé l'algorithme `ADD-SUB-TRI` sur quelques échantilons. Les résultats obtenus, décrits dans [BSS02e], nous ont amenés à compléter notre investigation par une approche qui est décrite dans la section suivante.

3.2 Etude combinatoire des configurations compatibles avec une distance additive

3.2.1 Configurations des familles de graphes instanciant une matrice ordinale additive

En examinant des données réelles et des données "additives" simulées, nous avons remarqué que certaines successions de graphes apparaissent beaucoup plus fréquemment que d'autres.

Nous avons donc mené une étude combinatoire pour préciser dans quelles conditions une matrice ordinale peut correspondre à une matrice additive, ce qui nous a permis de dégager des paramètres qui régissent le comportement des graphes définis à partir d'une distance arborée. Cette étude est à rapprocher de l'énumération réalisée dans [Gue98].

Nous examinons ici les différentes suites $G_0, ..., G_k$ de graphes emboîtés associés à une matrice ordinale. Compte tenu de ce qui a été dit dans le chapitre précédent, nous nous concentrons sur l'étude des configurations de graphes à 4 sommets associées à une distance additive. Une étude combinatoire complète présenterait un nombre de cas trop important pour qu'on puisse en tirer facilement des conclusions. Nous commençons donc par examiner dans quelles conditions nous pouvons éliminer certaines configurations en étudiant les propriétés liées à l'isomorphisme.

Définition 3.2.1 CONFIGURATION, TRANSITION.
Une famille de graphes emboîtés sera appelée ***configuration*** *quand elle correspondra à une matrice ordinale additive. La succession de deux graphes dans une configuration sera appelée* ***transition****.*

Une configuration étant définie pour une famille de graphes, elle sera définie pour un ensemble de matrices ordinales qui sont associées à cette même famille. Il faudra donc examiner ces configurations à une permutation des sommets près, ce qui nous amène à définir la notion d'**isomorphisme de famille**.

Définition 3.2.2 FAMILLES DE GRAPHES ISOMORPHES.
Soit $\mathcal{F} = G_0, ...G_k$ et $\mathcal{F}' = G'_0, ..., G'_k$ deux familles de graphes emboîtés sur les ensembles de sommets respectifs $V = \{x_1, ..., x_n\}$ et $V' = \{x'_1, ..., x'_n\}$. On dira que $\mathcal{F}$ et $\mathcal{F}'$ sont des familles isomorphes quand il existe un isomorphisme de graphes ψ commun à tous les graphes : $\forall i \in [0..k]$, $\psi(G_i) = \psi(G'_i)$.

Cette définition s'appuie sur la notion d'isomorphisme de graphes que nous rappelons :

Définition 3.2.3 GRAPHES ISOMORPHES.
Soient $G = (V, E)$ et $G' = (V', E')$ deux graphes. G et G' sont dits isomorphes quand il existe une bijection φ de V vers V' qui vérifie $\forall x, y \in V$, $xy \in E \Longleftrightarrow \varphi(x)\varphi(y) \in E'$.

Les graphes G et G' ont par conséquent le même nombre de sommets et le même nombre d'arêtes. Deux configurations isomorphes ont le même nombre de graphes emboîtés.

Quand deux familles sont isomorphes, les graphes correspondants sont deux à deux isomorphes, mais on n'a pas la réciproque : les graphes correspondants peuvent être deux à deux isomorphes sans qu'il puisse exister un isomorphisme commun qui permette de relier les familles (nous dirons alors que les deux familles sont "pseudo-isomorphes"). C'est ce qu'illustre l'exemple 3.2.4.

Exemple 3.2.4
La figure 3.6 présente quatre familles de graphes.

• La famille $\mathscr{H}$ est isomorphe à la famille $\mathscr{G}$. La permutation φ : $\{b \leftrightarrow d\}$ est un isomorphisme de graphes commun à tous les graphes : $\varphi(G_1) = H_1$, $\varphi(G_2) = H_2$), etc.

• La famille $\mathscr{H}'$ n'est pas isomorphe à la famille $\mathscr{G}$. Il existe deux isomorphismes (l'identité et la permutation $\{a \leftrightarrow c\}$) qui relient G_1 à H'_1, mais aucun de ces deux isomorphismes ne permet de relier G_3 à H'_3. G_3 est cependant isomorphe à H'_3 (permutation circulaire $\{a \leftrightarrow d, d \leftrightarrow c, c \leftrightarrow b, b \leftrightarrow a\}$) ; de même, G_2 et H'_2 sont isomorphes (permutation $\{a \leftrightarrow c\}$) ainsi que G_4 et H'_4 (permutation $\{a \leftrightarrow b\}$). Ces deux familles sont "pseudo-isomorphes".

• Le graphe H''_2 de la famille $\mathscr{H}''$ n'est pas isomorphe au graphe G_2, par conséquent les deux familles ne sont pas isomorphes ni même "pseudo-isomorphes".
◇

Nous allons maintenant définir l'ensemble des configurations possibles en utilisant comme représentation intermédiaire un ordre partiel sur les graphes à n sommets inclus dans une clique K_n. Cet ensemble forme un treillis des parties en considérant l'inclusion des graphes au sens des arêtes. Si nous ne retenons parmi ces graphes à n sommets qu'un seul représentant par ensemble de graphes isomorphes, nous obtenons un sous-ordre que nous noterons $\mathcal{P}_G(n)$, qui est donc l'ordre quotient du treillis par la relation d'isomorphisme. La figure 3.7 montre le treillis des parties du graphe K_3 et l'ordre $\mathcal{P}_G(n)$ associé.

Théorème 3.2.5 ORDRE $\mathcal{P}_G(n)$.
Pour toute valeur de n, l'ensemble $\mathcal{P}_G(n)$ des graphes à n sommets, définis à un isomorphisme près et ordonnés par inclusion d'arêtes, est un ordre gradué et ipsodual.

Preuve : Soit $\mathcal{P}_G(n)$ l'ensemble à un isomorphisme près des graphes sur un ensemble V de n sommets.

- Par inclusion des graphes au sens des arêtes, $\mathcal{P}_G(n)$ est un ordre ; par contraction de sommets isomorphes, c'est un sous-ordre du treillis des graphes inclus dans K_n.
- Soit γ la fonction de V vers IN qui à tout graphe de $\mathcal{P}_G(n)$ associe son nombre d'arêtes. Il est évident que si G est inférieur à G' dans $\mathcal{P}_G(n)$ alors $\gamma(G) < \gamma(G')$; de plus si G' couvre G alors $\gamma(G') = \gamma(G)+1$. γ constitue donc une graduation de $\mathcal{P}_G(n)$.
- Tout graphe $G = (V, E)$ de $\mathcal{P}_G(n)$ a un complémentaire qui est $\bar{G} = (V, V^2-E)$ et qui ne lui est pas comparable. De plus, si $G_1 = (V, E_1)$ est inférieur à $G_2 = (V, E_2)$ alors $E_1 \subset E_2$, $V^2-E_1 \subset V^2-E_2$ et donc $\bar{G}_2$ est inférieur à $\bar{G}_1$ dans $\mathcal{P}_G(n)$ et donc supérieur dans son dual. Par conséquent $\mathcal{P}_G(n)$ est ipsodual.

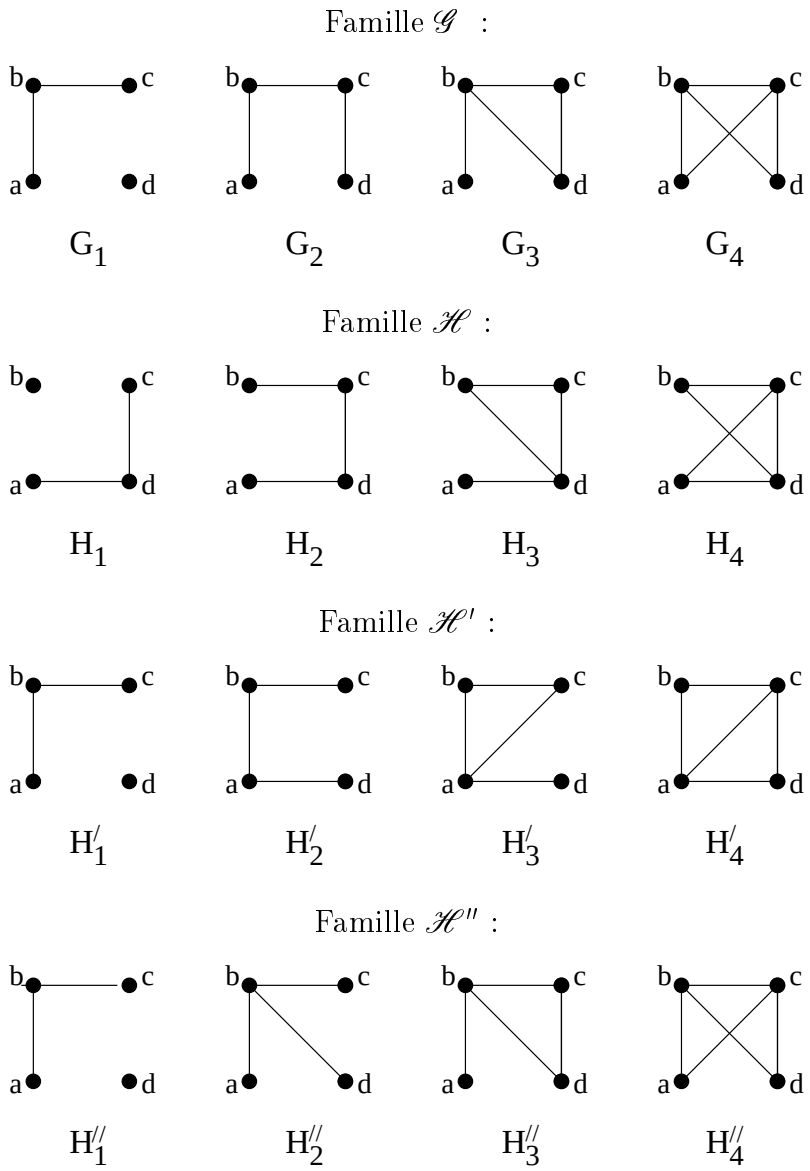


FIG. 3.6 – Les familles de graphes de l'exemple 3.2.4.

◇

Propriété 3.2.6
Pour $p < q$, $\mathcal{P}_G(p)$ est un sous-ordre de $\mathcal{P}_G(q)$

Preuve : En voici une preuve constructive : ajouter à chaque graphe de $\mathcal{P}_G(p)$ $q-p$ sommets isolés. Les différents graphes obtenus ont q sommets et ne sont pas deux à deux isomorphes, ce sont donc des éléments distincts de $\mathcal{P}_G(q)$.
◇

Propriété 3.2.7
Pour $n \leq 4$, $\mathcal{P}_G(n)$ est un treillis.

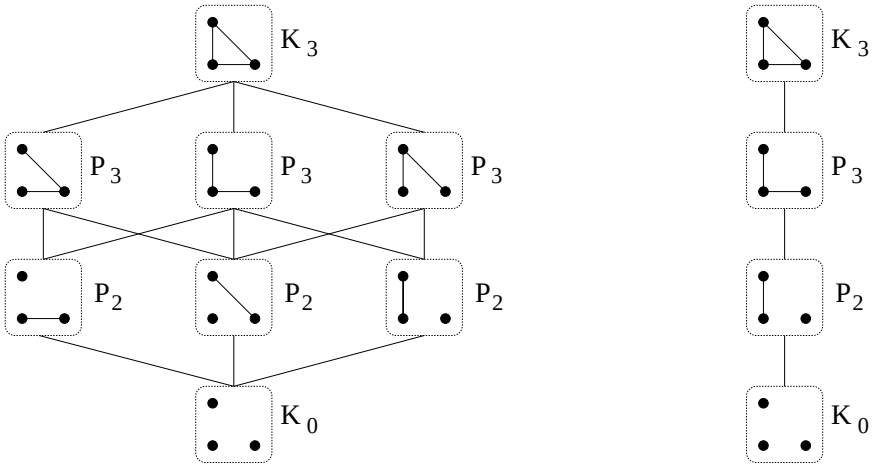


FIG. 3.7 – Le treillis des parties du graphe K_3 et l'ordre $\mathcal{P}_G(n)$ associé.

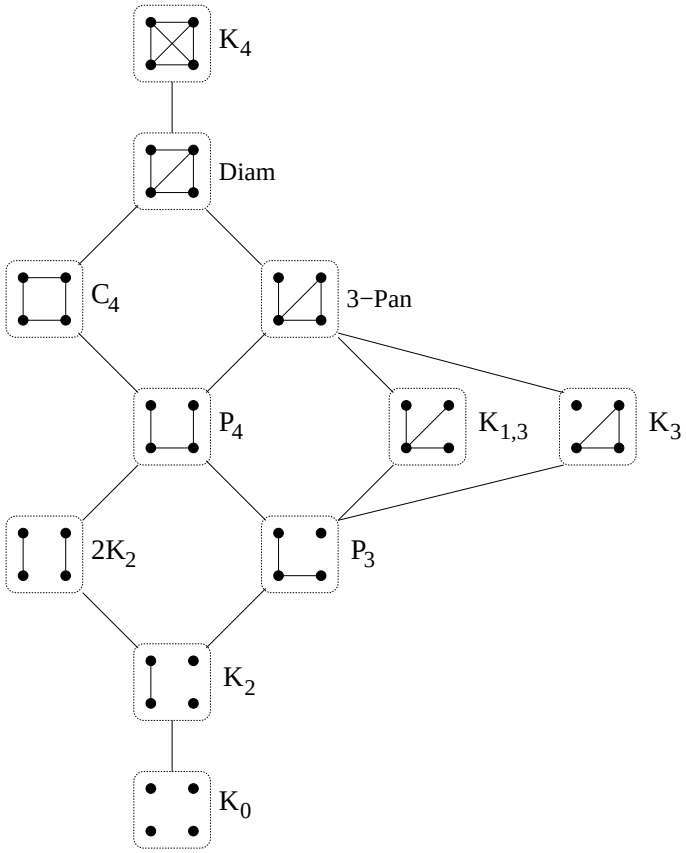


FIG. 3.8 – Le treillis $\mathcal{P}_G(4)$.

Pour $n > 4$ $\mathcal{P}_G(n)$ n'est pas un treillis.

Preuve :
• $\mathcal{P}_G(2)$ est composé de deux éléments : le stable à deux sommets et $K2$.
• $\mathcal{P}_G(3)$ est une chaîne, comme le montre la figure 3.7.
• Le treillis $\mathcal{P}_G(4)$ est présenté dans la figure 3.8.
• $\mathcal{P}_G(5)$ n'est pas un treillis, comme le montre la figure 3.9 : K_2 et P_3 n'ont pas de borne supérieure car Maj(K_2,P_3) a deux éléments minimaux K_2P_3 et P_4 et donc pas de minimum.
• La propriété 3.2.6, permet d'affirmer que le contre-exemple de K_2 et P_3 se retrouve dans tous les ordres de taille supérieure : $\mathcal{P}_G(n)$, $n > 5$.
◇

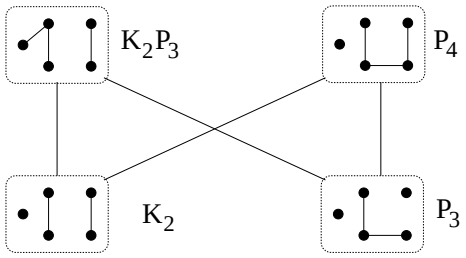


FIG. 3.9 – Une partie de l'ordre $\mathcal{P}_G(5)$.

L'ipsodualité peut s'avérer très intéressante car un treillis ipsodual est le treillis de Galois d'une relation symétrique qui est donc représentable par un graphe. Le travail de Berry et Bordat ([BB99]) a montré que, dans le cas des treillis orthocomplémentés (qui sont des treillis ipsoduaux particuliers) on pouvait attribuer une sémantique forte aux graphes construits sur le complémentaire d'une relation. En conséquence nous étudions le graphe G complémentaire de la relation $\mathcal{R}$ réduite associée à $\mathcal{P}_G(4)$.

Le procédé de construction de ce graphe est le suivant :

1. Le treillis $\mathcal{P}_G(4)$ a pour sup-irréductibles K_2, $2K_2$, P_3, $K_{1,3}$, K_3 et C_4 et pour inf-irréductibles $Diam$, C_4, 3-Pan, K_3, $K_{1,3}$ et $2K_2$.
2. On construit la relation $\mathcal{R}$ entre l'ensemble des sup-irréductibles et l'ensemble des inf-irréductibles telle que $(x, y) \in \mathcal{R}$ quand le sup-irréductible x est au-dessous de l'inf-irréductible y dans le treillis $\mathcal{P}_G(4)$. La table de cette relation est la suivante :

	K_2	$2K_2$	P_3	$K_{1,3}$	K_3	C_4
$Diam$	×	×	×	×	×	×
C_4	×	×	×			
3-Pan	×	×	×	×	×	
K_3	×		×			
$K_{1,3}$	×		×			
$2K_2$	×					

La table de cette relation est symétrique car le treillis est ipsodual ;
les inf–irréductibles $Diam$, C_4, 3-Pan, K_3, $K_{1,3}$ et $2K_2$ ont pour complémentaires respectifs les sup-irréductibles K_2, $2K_2$, P_3, $K_{1,3}$, K_3 et C_4.

3. On construit le graphe G en prenant le complémentaire de la relation et en remplaçant les inf-irréductibles par leurs complémentaires respectifs. La matrice d'incidence de G est la suivante :

	K_2	$2K_2$	P_3	$K_{1,3}$	K_3	C_4
K_2	0	0	0	0	0	0
$2K_2$	0	0	0	1	1	1
P_3	0	0	0	0	0	1
$K_{1,3}$	0	1	0	1	1	1
K_3	0	1	0	1	1	1
C_4	0	1	1	1	1	1

Le graphe G est présenté dans la figure 3.10.

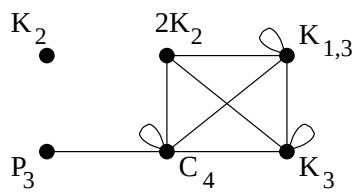


FIG. 3.10 – Le graphe G associé au treillis ipsodual $\mathcal{P}_G(4)$.

Les sommets Diam et $3-Pan$ sont représentés dans G par leurs complémentaires respectifs K_2 et P_3. Les boucles apparaissent pour des sommets qui, dans $\mathcal{P}_G(4)$, étaient des éléments à la fois sup-irréductibles et inf-irréductibles. P_4 n'apparaît pas dans G, il n'était pas un élément irréductible de $\mathcal{P}_G(4)$, ni le complémentaire d'un irréductible.

Dans le graphe de la figure 3.10, on peut remarquer que $K_{1,3}$ et K_3 forment un moplex : les deux sommets correspondants partagent le même voisinage et jouent donc exactement le même rôle. Cela peut se vérifier sur le treillis $\mathcal{P}_G(4)$ (voir figure 3.8) où $K_{1,3}$ et K_3 ont la même couverture et couvrent le même graphe. On peut donc prévoir que, dans le comportement des configurations, $K_{1,3}$ et K_3 joueront le même rôle.

3.2.2 Règles extraites de l'étude des configurations à 4 sommets

Au vu des résultats qui précèdent, pour établir des statistiques sur la fréquence d'un graphe triangulé donné dans une configuration ou sur la fréquence d'une transition entre deux graphes triangulés, il est suffisant de considérer les familles de graphes à un isomorphisme de famille près. De plus, notre analyse montre que, pour notre propos, il n'y a pas de différences significatives à considérer les familles à un isomorphisme de famille près ou à un isomorphisme de graphe près.

Nous nous sommes plus particulièrement intéressés aux configurations à quatre sommets, en cohérence avec la condition des quatre points. Nous avons vu, en effet, au chapitre précédent que toute matrice ordinale était instanciable en distance, il ne manque alors que la condition des quatre points pour obtenir une distance additive d'arbre.

Nous reprendrons les notations de [Gue98] : chaque configuration à quatre sommets

peut être définie par une suite (s_1, s_2, s_3, s_4, s_5, s_6) de six seuils qui correspondent aux six valeurs ab, cd, ac, bd, ad et bc d'une distance sur quatre objets $\{a, b, c, d\}$.

Nous présentons dans la figure 3.12 les configurations à 4 sommets $\{a, b, c, d\}$ établies à un isomorphisme de graphes près. Nous avons synthétisé les résultats dans un tableau, présenté dans la figure 3.11. Ce tableau indique, pour chaque couple de graphes à 4 sommets, le nombre de transitions dans lesquelles ces graphes se succèdent directement. La dernière ligne donne le nombre total d'occurrences de chaque graphe. La présence d'un tiret dans une case indique un cas impossible.

	K_2	P_3	$2K_2$	K_4	$K_{1,3}$	K_3	3-Pan	$Diam$	K_4
K_0	29	14	5	2	1	1	1	**0**	**0**
K_2	-	8	3	2	4	4	2	1	1
P_3	-	-	-	4	6	6	5	2	**0**
$2K_2$	-	-	-	4	-	-	2	**0**	2
P_4	-	-	-	-	-	-	6	6	**0**
$K_{1,3}$	-	-	-	-	-	-	7	3	1
K_3	-	-	-	-	-	-	7	3	1
3-Pan	-	-	-	-	-	-	-	15	13
$Diam$	-	-	-	-	-	-	-	-	31
Total	29	22	8	12	11	11	30	30	45

FIG. 3.11 – Nombre de transitions et nombre d'apparitions de chaque graphe, à partir de l'inventaire des configurations de la figure 3.12.

De ce tableau nous pouvons inférer plusieurs règles et statistiques :

Lemme 3.2.8 "RÈGLES".

1. *Aucune configuration ne contient de C_4 (ce n'est pas un graphe triangulé) ;*
2. *Pour chaque configuration ayant $K_{1,3}$ à une position i, il existe une configuration où $K_{1,3}$ est remplacé par K_3, et réciproquement ;*
3. *$2K_2$ n'est jamais suivi par Diam ;*
4. *Aucune configuration ne commence par Diam ni par K_4 ;*
5. *Ni P_3 ni P_4 ne peuvent être directement suivis de K_4 dans une configuration.*

Si la règle 1 n'est pas surprenante, puisque nous avons établi au chapitre précédent que tous les graphes correspondant à une matrice additive sont triangulés, les autres règles sont loin d'être évidentes et pourraient être exploitées avec profit.

Nous pouvons aussi remarquer que, conformément à ce qui avait été prédit au vu de la figure 3.10, K_3 et $K_{1,3}$ ont bien un comportement identique, ce qui donne à penser que l'examen plus poussé de ce graphe pourrait aboutir à des résultats intéressants.

Lemme 3.2.9 "STATISTIQUES".

1. *K_4 suit habituellement soit Diam (69% des cas) soit 3-Pan (29% des cas), soit un total de 98% ;*

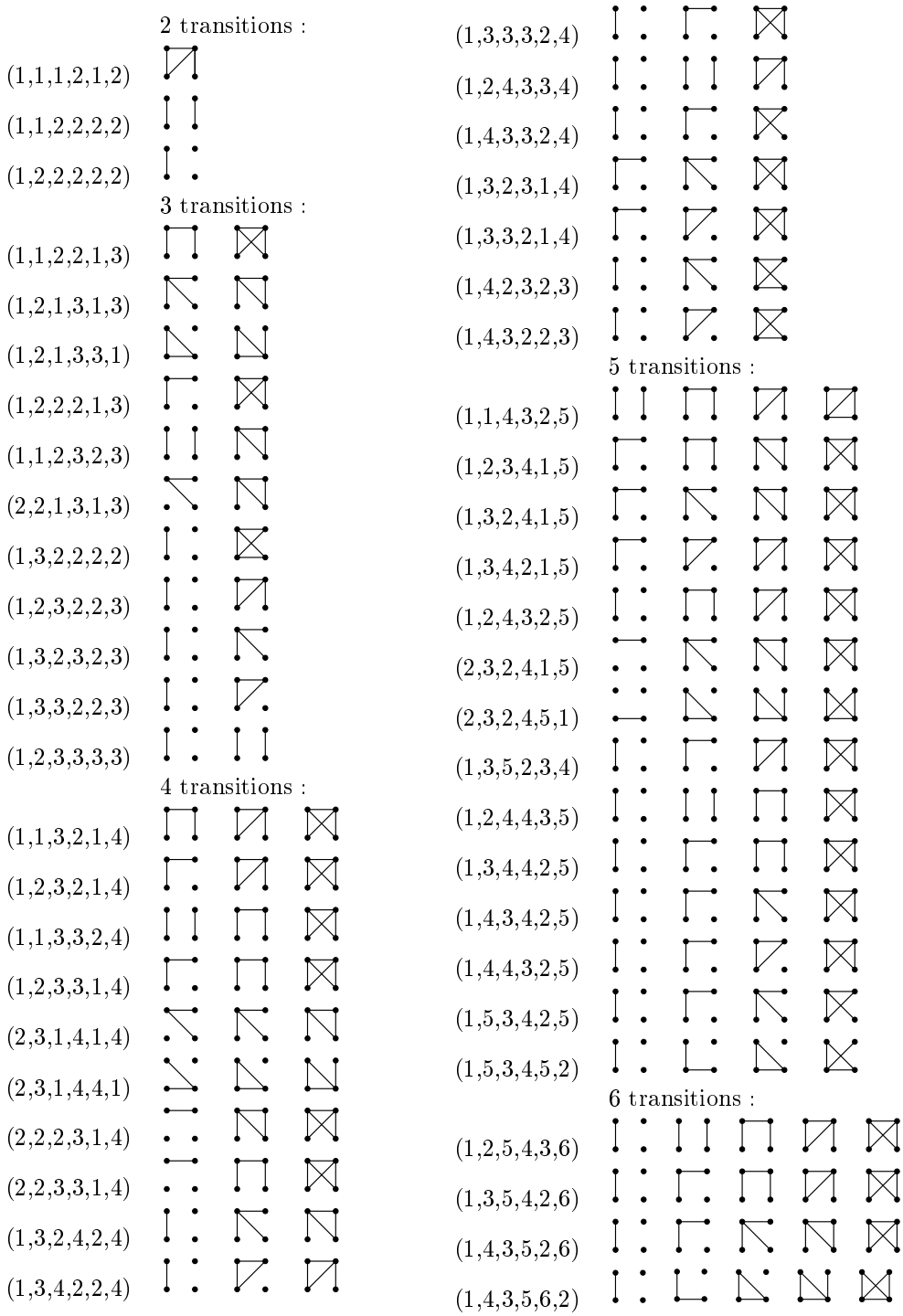


FIG. 3.12 – Inventaire des configurations à 4 sommets classées selon le nombre de transitions (le stable initial et la clique finale n'ont pas été représentés).

2. *$2K_2$ est le moins fréquent des graphes triangulés ;*
3. *$2K_2$ est plus souvent suivi de P_4 (50%) que des deux autres graphes possibles : 3-Pan et K_4 (25% chacun).*

Les résultats ci-dessus permettent d'apporter des idées de stratégies nouvelles pour trianguler tout en se rapprochant mieux d'une matrice additive.

Un des résultats les plus surprenants de l'analyse statistique est la très faible proportion de $2K_2$: 4%.

Ce résultat est intéressant car le complémentaire d'un graphe triangulé sans $2K_2$ est lui-même triangulé. Les graphes triangulés et co-triangulés sont appelés graphes split (*split graphs*) et forment une classe bien étudiée ayant des propriétés structurelles fortes.

De plus, en testant de nombreuses distances additives, nous avons vu que la plupart des $2K_2$ présents étaient dus au fait que les graphes n'étaient pas connexes : chaque fois qu'au moins deux composantes connexes comportent plusieurs arêtes cela fait apparaître un $2K_2$ dans le graphe global. Pour affiner notre approche nous sommes donc ramenés à l'étude de chaque composante connexe séparément. Ceci débouche sur la notion de graphe localement split.

Définition 3.2.10 GRAPHE LOCALEMENT SPLIT.
Un graphe triangulé sera dit ***localement split*** *(*locally split graph*) quand chacune de ses composantes connexes est un graphe split.*

Nous pouvons améliorer l'algorithme `ADD-SUB-TRI` en maintenant un graphe localement split ou presque localement split.

3.3 Perspectives

La matrice d'une distance triangulée peut être décomposée en utilisant le procédé de décomposition d'un graphe par séparateurs minimaux complets. Cette décomposition induit une décomposition de la phylogénie associée. La décomposition peut être réalisée même avec un séparateur minimal non complet et sur un graphe qui n'est pas triangulé.

On peut illustrer ce procédé avec des données issues de [GG00] qui utilise une phylogénie inférée à partir d'une dissimilarité non triangulée. La figure 3.13 montre la phylogénie inférée et le graphe G_{35} correspondant au seuil n°35 de la dissimilarité. G_{35} n'est pas triangulé : il possède un cycle $abfg$ sans corde. On voit que la struture de G_{35} reflète celle de la phylogénie : à la partition des sommets de G_{35} en $\{a, b, c, d, e, f, g, n, o\}$ et $\{h, i, j, k, l, m\}$) correspond une partition de la phylogénie en deux sous-arbres.

En choisissant le séparateur minimal non complet $\{a, d, e, f, o\}$ du graphe G_{35}, on peut décomposer ce graphe en deux sous-graphes $G' = G_{35}(\{b, c, d, e, f, n, o\})$ et $G'' = G_{35}(\{a, g, h, i, j, k, l, m\})$ (figure 3.14), les éléments du séparateur minimal $\{a, d, e, f, o\}$ étant répartis entre ces deux graphes. On retrouve cette décomposition au niveau de la phylogénie inférée (figure 3.15).

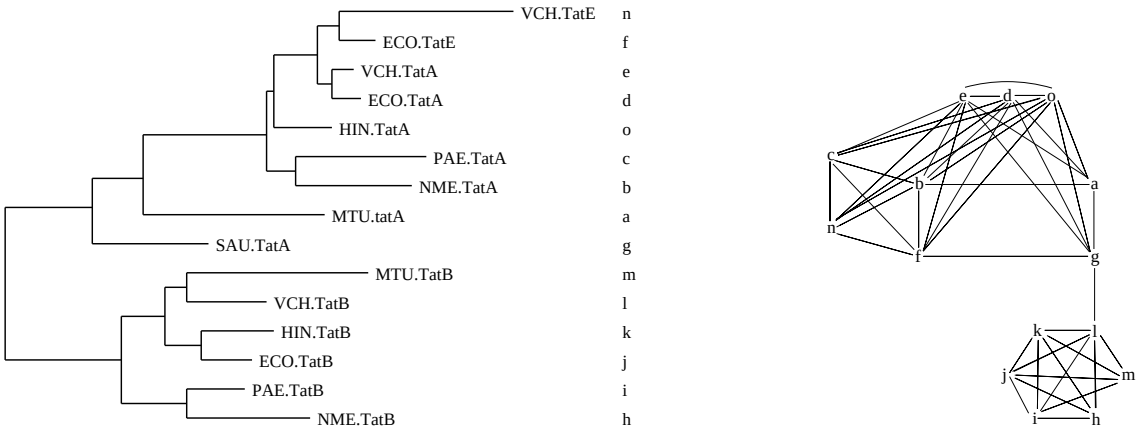


FIG. 3.13 – La phylogénie de [GG00] et le graphe non triangulé G_{35}.

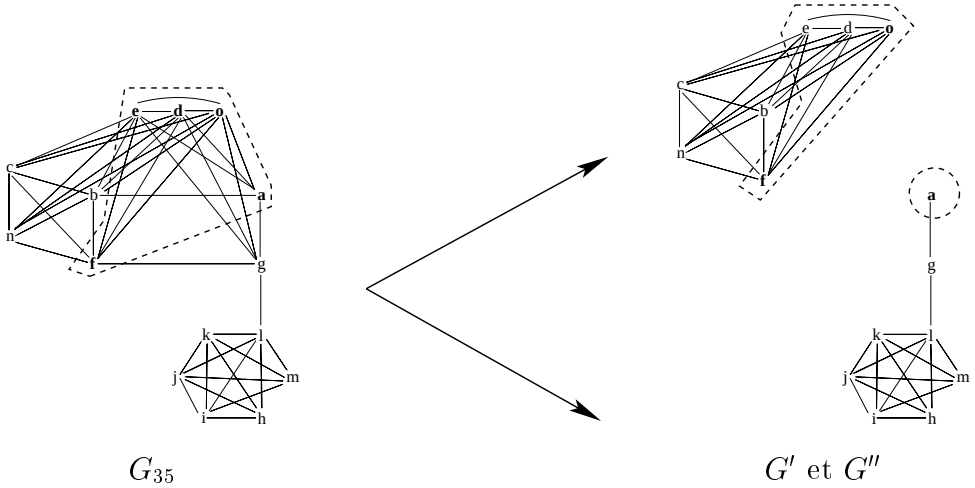


FIG. 3.14 – Décomposition de G_{35} par le séparateur non complet $\{a, d, e, f, o\}$.

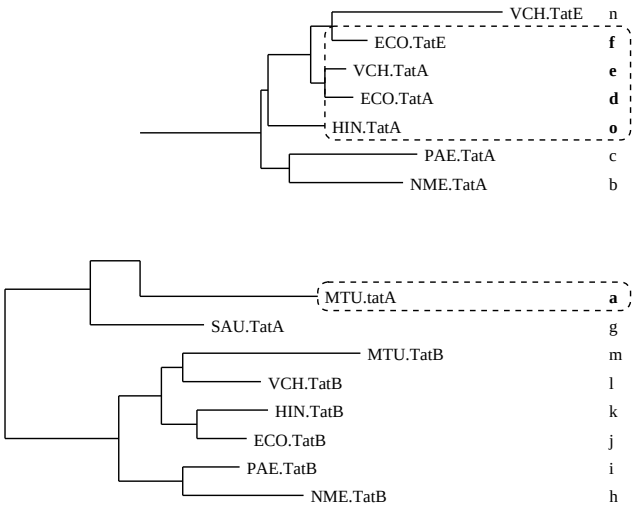


FIG. 3.15 – Décomposition de la phylogénie selon le séparateur $\{a, d, e, f, o\}$ de G_{35}.

Conclusion de la première partie

Nous avons, à partir d'un problème issu de l'univers biologique, mis en place une problématique de graphes nouvelle et nous avons défini le concept de sous-triangulation maximale.

L'étude des configurations de graphes ouvre des voies nouvelles, tant du point de vue théorique avec, par exemple, l'étude des graphes définis par les ordres ipsoduaux, que du point de vue expérimental avec l'exploitation des règles et statistiques établies au chapitre 3.

Bien que ce travail soit loin d'être terminé, les résultats sont prometteurs. Un projet franco-israélien est en cours de mise en place avec le professeur Golumbic, dans le but de poursuivre à la fois l'exploration théorique et l'investigation expérimentale des résultats décrits dans cette première partie.

Deuxième partie

ANALYSE FORMELLE DE CONCEPTS

Introduction de la deuxième partie

Explorer rapidement une base de données, y découvrir des motifs, tels sont les enjeux de la science émergente appelée "Data Mining", "fouille de données", ou encore "extraction de données".

Une des approches est d'explorer toutes les combinaisons possibles offertes par cette base de données, afin d'examiner quelles seraient les plus prometteuses ; cela s'avère extrêmement difficile, bien évidemment, en raison du nombre exponentiel de ces combinaisons. Pourtant, cette méthode est encore utilisée dans bien des applications.

Une approche beaucoup plus efficace consiste à factoriser certains attributs ou propriétés de façon, d'une part, à diminuer le nombre de combinaisons pertinentes et, d'autre part, à structurer les données de façon à extraire un aspect sémantique.

Une structure très prometteuse et très étudiée ces dernières années est celle du treillis de Galois, ou treillis des concepts. Elle reste, certes, de taille exponentielle dans le cas général, mais elle est nettement plus petite et surtout plus sémantique que le simple treillis des parties.

Nous nous sommes attachés à l'étude de cette structure. Nous avons découvert que ces treillis se décrivaient bien par un graphe, pour ensuite mettre en évidence une notion nouvelle d'ordre sur les sommets de ce graphe, associé à la notion classique de domination dans un graphe. Cet ordre semble décrire de façon à la fois très efficace et très fidèle les processus mis en jeu dans les algorithmes de gestion d'un treillis des concepts.

Chapitre 4

Codage d'un treillis par un graphe

De par sa taille, un treillis des concepts s'avère souvent difficile à gérer. Il est donc tentant de lui trouver une représentation de faible taille, facile à manipuler, qui constitue un codage du treillis, dans le sens où elle permet d'effectuer directement des opérations comme le calcul des bornes supérieures et inférieures de plusieurs éléments, la vérification qu'un rectangle est un concept, le calcul de la fermeture d'une partie, et ainsi de suite. Il existe plusieurs codages d'un treillis ; en particulier la relation binaire réduite en est un, ainsi que le graphe biparti orienté construit sur la relation, utilisé par [Bor92] et [MN93].

Notre approche est différente, puisque nous travaillons sur des graphes non-orientés. Elle s'inspire des résultats sur le treillis de séparabilité ([Ber95], [BB99]), où il est démontré que si l'on travaille sur une relation symétrique (qui représente donc un graphe), les rectangles maximaux du complémentaire de la relation définissent des séparateurs du graphe associé. Nous allons ainsi définir ici un graphe, construit sur le complémentaire de la relation, qui nous permettra de la même façon de travailler avec des séparateurs, pour lesquels on dispose d'une "boîte à outils" bien fournie.

Dans ce chapitre, nous introduisons les outils théoriques de base qui constituent l'originalité de notre approche.

Dans une première section, nous définissons un graphe $G_{\mathcal{R}}$ construit à partir d'une relation binaire $\mathcal{R}$ quelconque, nous étudions ses propriétés, et en particulier montrons l'équivalence entre un séparateur minimal de ce graphe et un concept du treillis $\mathcal{L}(\mathcal{R})$ construit sur $\mathcal{R}$.

Dans une deuxième section, nous étudions l'effet de la saturation d'un séparateur minimal sur le treillis résultant, en établissant le lien entre séparateurs non croisants et concepts comparables ; nous verrons que cela nous permet notamment de calculer des sous-treillis de $\mathcal{L}(\mathcal{R})$.

La troisième section applique au treillis les décompositions d'un graphe par séparateurs minimaux, qui nous permettront au chapitre 6 de travailler récursivement sur des sous-relations.

Pour terminer, nous tirons des résultats présentés des éléments nouveaux permettant

de discuter de la taille – exponentielle ou polynomiale – du treillis par rapport à celle de la relation.

Pour ce chapitre, deux papiers ont été écrits en collaboration avec Anne Berry :

- Les résultats sur le codage d'un treillis par un graphe ont été présentés et publiés dans le workshop "Discrete Mathematics and Data Mining" organisé par Peter Hammer dans le cadre de la deuxième conférence internationale de SIAM sur le Data Mining ([BS02a]), et un article a été soumis pour le numéro spécial de Discrete Applied Mathematics correspondant.
- Les résultats sur la polynomialité d'un treillis ont été présentés et publiés dans le workshop "Formal Concept Analysis for Knowledge Discovery in Data Bases", organisé par Michel Liquière dans la conférence européenne sur l'intelligence artificielle ECAI'02 ([BS02b]), et sont en cours d'implémentation pour présentation en version étendue.

4.1 Un nouveau codage du treillis des concepts

4.1.1 Graphe co-biparti associé à une relation

Nous présentons tout d'abord le graphe co-biparti que nous construisons à partir de la relation et nous montrons quelques propriétés des co-bipartis qui nous seront utiles par la suite.

Définition du graphe sous-jacent

Définition 4.1.1 GRAPHE SOUS-JACENT À UN CONTEXTE.
*Soit $C = (\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte. Nous définissons le graphe sous-jacent à C (*underlying graph*), noté $G_{\mathcal{R}}$, de la façon suivante :*
- *L'ensemble des sommets de $G_{\mathcal{R}}$ est $V = \mathscr{P} \cup \mathscr{O}$;*
- *$\mathscr{P}$ et $\mathscr{O}$ sont des cliques ;*
- *Pour un sommet x de $\mathscr{P}$ et un sommet y de $\mathscr{O}$, il y a une arête xy dans G si et seulement si (x, y) n'appartient **pas** à $\mathcal{R}$.*

Nous noterons $n = |V| = |\mathscr{P}| + |\mathscr{O}|$.

Remarquons que, dans $G_{\mathcal{R}}$, comme $\mathscr{P}$ et $\mathscr{O}$ sont des cliques, seules les arêtes entre un sommet de $\mathscr{P}$ et un sommet de $\mathscr{O}$ sont pertinentes. Les parcours de graphe n'utiliseront que ces arêtes, que nous qualifierons d'**externes** ; par conséquent, le nombre m d'arêtes de $G_{\mathcal{R}}$ fera par la suite référence à $|\bar{R}|$.

Nous utiliserons les notations suivantes :

- Pour une propriété $x \in \mathscr{P}$, $N^+(x) = N(x) \cap \mathscr{O}$, c'est-à-dire $N^+(x) = \{y \in \mathscr{O} | xy \in G_{\mathcal{R}}\} = \{y \in \mathscr{O} | (x,y) \notin \mathcal{R}\}$;
- Pour un objet $y \in \mathscr{O}$, $N^+(y) = N(y) \cap \mathscr{P}$, c'est-à-dire $N^+(y) = \{x \in \mathscr{P} | xy \in G_{\mathcal{R}}\} = \{x \in \mathscr{P} | (x,y) \notin \mathcal{R}\}$.

D'autre part, il suffit que $\mathcal{R}$ ne soit pas trivialement un rectangle de uns pour que $G_{\mathcal{R}}$ soit connexe ; de façon similaire, il suffit que $\mathcal{R}$ ne soit pas trivialement un rectangle de zéros pour que $G_{\mathcal{R}}$ ne soit pas une clique. Nous pouvons éliminer ces deux cas particuliers pour la suite de ce travail car ils sont faciles à détecter et ne présentent pas d'intérêt autre que théorique.

Par construction, le graphe sous-jacent à un contexte appartient à la classe des graphes co-bipartis (voir définition 1.3.11). Les graphes de cette classe jouissent de propriétés remarquables, qui en font une classe particulièrement facile à manipuler.

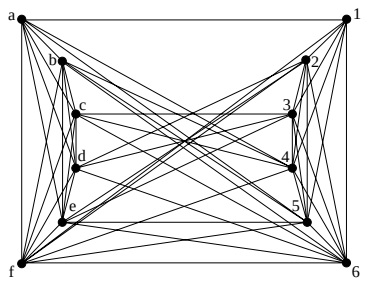


FIG. 4.1 – Graphe $G_{\mathcal{R}}$ de la relation $\mathcal{R}$ de l'exemple 4.1.2.

Exemple 4.1.2
Reprenons le contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ de l'exemple 1.2.51 (page 33, le treillis associé est à nouveau représenté dans la figure 4.2). La relation $\mathcal{R}$ était définie par la table suivante :

$\mathcal{R}$	a	b	c	d	e	f
1		x	x	x	x	
2	x	x	x			
3	x	x				x
4				x	x	
5			x	x		
6	x					

Le graphe sous-jacent à ce contexte est le graphe $G_{\mathcal{R}}$ de la figure 4.1.
$G_{\mathcal{R}}$ n'a ni sommet universel ni sommet simplicial. $N^+(a) = \{1, 4, 5\}$ et $N^+(1) = \{a, f\}$.
$\diamond$

Sous-graphes d'un co-biparti

Propriété 4.1.3 HÉRÉDITÉ.
Tout sous-graphe d'un graphe co-biparti est co-biparti.

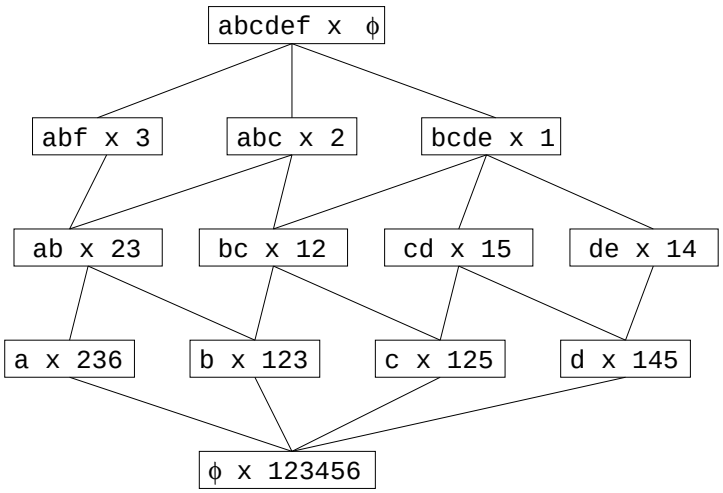


FIG. 4.2 – Treillis des concepts $\mathcal{L}(\mathcal{R})$ associé à la relation de l'exemple 4.1.2

Preuve : Soit $G = (V, E)$ un graphe co-biparti défini sur les cliques X et Y, $V = X+Y$. Soit x un sommet de V et soit $G(V-\{x\})$ le sous-graphe obtenu en retirant x de G. On peut supposer, sans perte de généralité, que x appartient à X. Si $X-\{x\}$ est vide alors $G(V-\{x\})$ est une clique qui est un graphe co-biparti (puisqu'un stable est un graphe biparti). Sinon, $G(V-\{x\})$ est défini sur les cliques $X-\{x\}$ et Y avec les mêmes non-arêtes reliant des sommets de ces deux ensembles que dans G ; son complémentaire sera défini sur les stables $X-\{x\}$ et Y avec un certain nombre d'arêtes entre ces deux ensembles, ce qui définit bien un biparti.
◇

Nous pourrons ainsi définir des sous-graphes qui seront automatiquement des co-bipartis et correspondront donc à des sous-relations.

Triangulations minimales d'un co-biparti

Propriété 4.1.4 SANS STABLE.
Un graphe co-biparti n'a pas de stable de taille supérieure à deux.

Preuve : Supposons qu'il existe un graphe co-biparti qui ait un stable de taille trois ou plus. Alors la partition des sommets du co-biparti en deux cliques oblige à mettre (au moins) deux sommets du stable dans une même clique ce qui est contradictoire.
◇

Corollaire 4.1.5 CLAW-FREE.
Un graphe co-biparti est Claw-free (c'est-à-dire sans $K_{1,3}$).

Preuve : Comme un $K_{1,3}$ possède un stable de taille trois donc de taille supérieure à deux, il ne peut être co-biparti (propriété 4.1.4), et par contraposition de la propriété 4.1.3, tout graphe ayant un $K_{1,3}$ pour sous graphe ne peut être co-biparti.
◇

Corollaire 4.1.6 AT-FREE.
Tout graphe co-biparti est AT-free (c'est-à-dire sans triplet astéroïdal).

Preuve : Par la propriété 4.1.4, un co-biparti n'a pas de stable de taille supérieure à deux, et ne peut donc pas avoir de triplet astéroïdal qui est un stable particulier.
◇

Ces corollaires assurent un comportement remarquable pour la triangulation des graphes co-bipartis :

Caractérisation 4.1.7 *[Par96]).*
Un graphe G est AT-free et Claw-free si et seulement si toute triangulation minimale de G est un graphe d'intervalles propres.

Propriété 4.1.8 *([Mei00]).*
Le calcul d'une triangulation minimale d'un graphe Claw-free et AT-free peut être fait en temps linéaire.

De ce qui précède, nous pouvons déduire que toute triangulation minimale d'un graphe co-biparti est un graphe d'intervalles propres, qui peut se calculer en temps linéaire. Rappelons que, pour un graphe quelconque, le calcul d'une triangulation minimale requiert un temps en $O(nm)$.

Ce résultat est important car nous verrons par la suite que les séparateurs minimaux complets jouent un rôle important ; or, comme nous l'avons vu au Chapitre 1, la façon la plus simple de trouver l'ensemble de ces séparateurs complets est de calculer d'abord une triangulation minimale du graphe.

Composantes connexes

Une autre propriété très importante, qui rend les co-bipartis particulièrement faciles à manipuler, est que chaque séparateur minimal définit exactement deux composantes connexes.

Propriété 4.1.9
Soit $G = (V, E)$ un graphe co-biparti construit sur les cliques X et Y, $V = X+Y$. Tout séparateur minimal S de G induit exactement deux composantes connexes A et B, l'une contenant uniquement des sommets de X et l'autre uniquement des sommets de Y.

Preuve : Soit $G = (V, E)$ un graphe co-biparti construit sur les cliques X et Y, $V = X+Y$, et S un séparateur minimal de G.
On sait, par la propriété 1.3.28, qu'un séparateur définit au moins deux composantes connexes A et B. Supposons que S définisse au moins les trois composantes A, B et C. Considérons trois sommets $x \in A$, $y \in B$ et $z \in C$. Ces trois sommets, puisqu'ils sont séparés par S, forment un stable de taille trois dans G, ce qui contredit la propriété 4.1.4. Par conséquent, S induit exactement deux composantes.
Comme A et B sont des composantes connexes induites par un séparateur minimal, elles ne sont pas vides. On peut supposer sans perte de généralité que A contient au moins un sommet de X. Puisque Y est une clique et que A et B sont deux composantes connexes distinctes de $G(V-S)$, B ne peut contenir de sommet de X ; par conséquent B contient seulement des sommets de Y. On montre suivant le même principe que $A \subseteq X$.
◇

En conséquence de cette propriété, les deux composantes connexes sont des composantes pleines, ce qui permet de les caractériser par leur voisinage.

Caractérisation 4.1.10
Soit $G = (V, E)$ un graphe co-biparti construit sur les cliques X et Y, $V = X+Y$, et S un ensemble de sommets de G. S est un séparateur minimal de G si et seulement si $G(V-S)$ a exactement deux composantes connexes A et B telles que $N(A) = N(B) = S$.

Preuve : La conclusion est immédiate en appliquant la propriété 1.3.28 à un graphe co-biparti, pour lequel la propriété 4.1.9 est vraie.
◇

4.1.2 Codage d'un treillis des concepts

Nous allons maintenant montrer que le graphe $G_{\mathcal{R}}$ que nous avons introduit en début de chapitre constitue un codage intéressant de la relation.

Equivalence entre concept d'un treillis et séparateur minimal du graphe sous-jacent

Nous allons commencer par énoncer le théorème suivant, qui est fondamental pour la suite de nos travaux :

Théorème 4.1.11 CONCEPT ET SÉPARATEUR MINIMAL.
Soient $C = (\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte, $G_{\mathcal{R}} = (V, E)$ le graphe co-biparti sous-jacent. $A \times B$ est un concept de C avec $A \neq \emptyset$, $B \neq \emptyset$ et $A \cup B \neq V$, si et seulement si $S = V-(A \cup B)$ est un séparateur minimal de $G_{\mathcal{R}}$.

Preuve : Soit $C = (\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte, $G_{\mathcal{R}} = (V, E)$ le graphe co-biparti sous-jacent.

$\Longrightarrow$

Soit $A \times B$ un concept de C avec $A \neq \emptyset$, $B \neq \emptyset$ et $A \cup B \neq V$; soit $S = V - (A \cup B)$. $S = V - (A \cup B)$ n'est pas vide. On peut affirmer que pour tous $a \in A$ et $b \in B$, S est un ab-séparateur minimal de $G_{\mathcal{R}}$. Tout d'abord, S est un ab-séparateur : s'il y avait une arête ab dans $G_{\mathcal{R}}$ alors par définition (a, b) ne serait pas dans $\mathcal{R}$ et $A \times B$ ne serait pas un concept. Nous allons ensuite prouver que S est minimal : supposons que ce ne soit pas le cas, par la propriété 1.3.28, on peut affirmer sans perte de généralité qu'il existe un sommet x de S qui ne voit aucun sommet de B, ce qui signifie que $\forall y \in B$, $(x, y) \in \mathcal{R}$; $Ax \times B$ serait alors un rectangle de $\mathcal{R}$ ce qui contredit la minimalité du rectangle $A \times B$.

$\Longleftarrow$

Soit S un séparateur minimal de $G_{\mathcal{R}}$ induisant les composantes connexes A et B. En tant que composantes connexes, A et B ne sont pas vides, en tant que séparateur, $S = V - (A \cup B)$ n'est pas vide ($\mathcal{R}$ étant supposée non complète), on a donc $A \cup B \neq V$. Puisque $\forall x \in A$, $\forall y \in B$, $xy \notin E$ et qu'ainsi $(x, y) \in \mathcal{R}$, on peut conclure que $A \times B$ est un rectangle de $\mathcal{R}$. Supposons que $A \times B$ ne soit pas minimal, sans perte de généralité, $\exists x \in \mathscr{O} - B$, $\forall y \in A$, $(y, x) \in \mathcal{R}$. Ainsi $x \in S$ et, dans $G_{\mathcal{R}}$, s ne verra aucun sommet de A, donc, par la propriété 1.3.28, S n'est pas minimal, ce qui est contradictoire.

$\diamond$

Remarque 4.1.12 *Avec les contraintes $A \neq \emptyset$, $B \neq \emptyset$ et $A \cup B \neq V$, non seulement on évacue le cas des relations complètes, mais aussi on écarte les éléments $\perp = \emptyset \times \mathscr{O}$ et $\top = \mathscr{P} \times \emptyset$, situations où aucun séparateur minimal ne peut être défini.*

Dans le cas d'une relation où un ensemble X de propriétés correspondrait à des colonnes de uns, le concept $\perp = X \times \mathscr{O}$ serait aussi associé à un séparateur minimal $\mathscr{P} - X$ du graphe sous-jacent.

Dans la cas d'une relation où un ensemble X de propriétés correspondrait à des colonnes de zéros, X serait, comme on le précisera dans la sous-section suivante, présent dans tous les séparateurs minimaux du graphe sous-jacent.

Les mêmes observations peuvent être faites sur les objets.

Pour $A \times B$ concept d'une relation $\mathcal{R}$ avec $S = V - (A \cup B)$, nous dirons que le séparateur minimal S **représente** le concept $A \times B$.

Nous pouvons maintenant reformuler la caractérisation 4.1.10 sur les graphes co-bipartis comme suit :

Caractérisation 4.1.13
Soient $C = (\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte, $G_{\mathcal{R}} = (V, E)$ le graphe sous-jacent et $A \times B$ un rectangle de $\mathcal{R}$ avec $A \neq \emptyset$, $B \neq \emptyset$ et $A \cup B \neq V$. $A \times B$ est un concept de C si et seulement si, dans $G_{\mathcal{R}}$, $N(A) = N(B)$.

De ceci, il résulte que la seule donnée de l'intension ou bien de l'extension d'un concept suffit pour inférer facilement les deux parties du concept.

Du théorème 4.1.11 nous pouvons déduire qu'un graphe co-biparti a un nombre de séparateurs minimaux potentiellement exponentiel, puisque un treillis des concepts peut avoir un nombre exponentiel d'éléments. Rappelons que, pour une taille donnée de l'ensemble $\mathscr{P}$, le plus gros treillis que l'on puisse obtenir est le treillis des parties $\mathscr{P}$ et que la relation associée à ce treillis a exactement un zéro dans chaque ligne et un zéro dans chaque colonne (dans ce cas, on a bien sûr $|\mathscr{P}| = |\mathscr{O}|$). Le graphe co-biparti sous-jacent à une telle relation, avec un nombre maximal de séparateurs minimaux, est donc le graphe où $|\mathscr{P}| = |\mathscr{O}|$ et chaque sommet de $\mathscr{P}$ voit exactement un sommet de $\mathscr{O}$.

Exemple 4.1.14
Dans la figure 4.3, $S = \{a, d, e, f, 3, 4, 5, 6\}$ est un séparateur minimal du graphe $G_{\mathcal{R}}$ de l'exemple 4.1.2. Il sépare $C_1 = \{b, c\}$ de $C_2 = \{1, 2\}$ et $bc \times 12$ est un concept de $\mathcal{R}$ et un élément de $\mathcal{L}(\mathcal{R})$ (voir exemple 4.1.2). Dans $G_{\mathcal{R}}$, $N(\{b, c\}) = N(\{1, 2\}) = S$.
$\diamond$

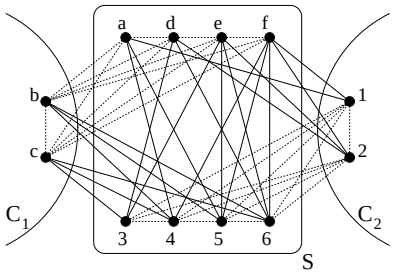


FIG. 4.3 – Séparateur S et composantes C_1 et C_2 de $G_{\mathcal{R}}$.

Borne supérieure ou inférieure d'un ensemble d'éléments

Etant donnés deux séparateurs minimaux de $G_{\mathcal{R}}$, on peut facilement trouver la borne supérieure et la borne inférieure des concepts qu'ils représentent.

Propriété 4.1.15
Soient $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte, $G_{\mathcal{R}} = (V, E)$ le graphe sous-jacent et $\mathcal{L}(\mathcal{R})$ le treillis des concepts associé. Soient S_1 et S_2 deux séparateurs minimaux de $G_{\mathcal{R}}$ représentant respectivement les concepts $A_1 \times B_1$ et $A_2 \times B_2$. Si on note $Y = S_1 \cup S_2$, $J = (V - Y) \cap \mathscr{O}$ est l'extension de Sup($A_1 \times B_1$,$A_2 \times B_2$) et $M = (V - Y) \cap \mathscr{P}$ est l'intension de Inf($A_1 \times B_1$,$A_2 \times B_2$).

Comme $S_1 = V - (A_1 \cup B_1)$ et $S_2 = V - (A_2 \cup B_2)$, ce qui précède découle directement de la propriété suivante :

Propriété 4.1.16 *([Bir67]).*
Soient $A_1 \times B_1$ et $A_2 \times B_2$ deux éléments d'un treillis des concepts. Alors $B_1 \cup B_2$ est l'extension de Sup($A_1 \times B_1$,$A_2 \times B_2$) et $A_1 \cup A_2$ est l'intension de Inf($A_1 \times B_1$,$A_2 \times B_2$).

4.1.3 Sommets universels et sommets simpliciaux

Etant donnée une relation, il se pose le problème de savoir si l'on travaille ou non sur une relation réduite. En général, la réduction d'une relation coûte $O(nm)$ (m étant le nombre de croix dans la relation, et n le nombre de propriétés et d'objets) ; dans les algorithmes que nous présenterons aux chapitres 5 et 6, nous sommes en-dessous de ce seuil pour chaque relation ou sous-relation traitée ; nous ne pourrons donc pas nous permettre de calculer la relation réduite. Par contre, nous pouvons réduire partiellement la relation, car il est possible de tester en temps linéaire les trois points suivants : existence de lignes ou de colonnes identiques, présence de rangées de zéros, présence de rangées de uns.

Nous discuterons donc ici de ces rangées de zéros et de uns, que nous pouvons détecter et éliminer. Nous allons voir que le retrait des sommets correspondants préserve la **structure articulatoire** d'un graphe, c'est-à-dire maintient l'ensemble des séparateurs minimaux inchangé ou produit une transformation uniforme des séparateurs minimaux. L'équivalence entre concept d'un treillis et séparateurs minimaux du graphe sous-jacent, que nous venons de montrer, permettra alors d'établir que le retrait des rangées de zéros et de uns dans une relation conserve le treillis des concepts.

Sommets universels d'un graphe

Propriété 4.1.17
Dans un graphe connexe, un sommet est universel si et seulement si il appartient à ***tous*** *les séparateurs minimaux du graphe.*

Preuve :

$\Longrightarrow$

Soit $G = (V, E)$ un graphe et x un sommet universel de G. Soit S un séparateur minimal quelconque de G. Soient C_1 et C_2 deux composantes connexes induites par S. Par la caractérisation 1.3.24, il n'y a aucune arête entre un sommet de C_1 et un sommet de C_2. Par conséquent x, qui est universel, ne peut appartenir ni à C_1 ni à C_2, ni plus généralement à aucune des composantes connexes induites par S ; donc x appartient à S.

$\Longleftarrow$

Soit $G = (V, E)$ un graphe connexe et x un sommet qui n'est pas universel. Il existe donc $y \in V$ non voisin de x. Puisque le graphe est connexe, il existe un chemin entre x et y, donc un xy-séparateur et donc un xy-séparateur minimal S. Comme x n'appartient pas à S, il ne peut pas appartenir à tous les séparateurs minimaux de G.

$\diamond$

Nous pouvons en déduire la propriété suivante.

Propriété 4.1.18
Soient $G = (V, E)$ un graphe connexe, U l'ensemble de ses sommets universels et S un sous-ensemble propre de V. S est un séparateur minimal de $G(V-U)$ si et seulement si $S \cup U$ est un séparateur minimal de G.

Preuve : Cette propriété est une conséquence directe des trois lemmes suivants. Le premier lemme garantit à la fois l'existence d'un schéma d'élimination des sommets universels et l'existence d'un schéma de recomposition du graphe par réinsertion de ces sommets universels, schémas qui ont tout deux pour invariant le fait que le sommet retiré ou ajouté reste universel dans le graphe intermédiaire considéré. Le second lemme affirme le maintien des séparateurs minimaux par ajout d'un sommet universel. Le troisième lemme affirme le maintien des séparateurs minimaux par retrait d'un sommet universel.
◇

Lemme 4.1.19
Soit $G = (V, E)$ un graphe et U l'ensemble des sommets universels de G. Alors pour toute partie T de U, les sommets de $U-T$ sont universels dans $G(V-T)$.

Preuve : Si un sommet u de $U-T$ est relié à tous les sommets de G, il reste relié à tous les sommets de $G(V-T)$ après retrait des sommets de T et donc universels dans $G(V-T)$.
◇

Lemme 4.1.20
Soit G un graphe et S un séparateur minimal de G. Le graphe obtenu en ajoutant un sommet universel u à G a notamment pour séparateur minimal $S \cup \{u\}$.

Preuve : Soit G un graphe et S un séparateur minimal de G. Supposons qu'il existe deux sommets x et y tels que S soit un xy-séparateur minimal de G. Tout d'abord, S est un séparateur de G donc, par la caractérisation 1.3.24, tout chemin reliant x et y passe par au moins un sommet de S. Après ajout de u, tout chemin reliant x et y passe toujours par $S \cup \{u\}$ puisque, pour ce qui concerne x et y, on a seulement ajouté le chemin xuy. Donc $S \cup \{u\}$ est un séparateur du graphe obtenu en ajoutant le sommet universel u au graphe G. Ensuite, par la même caractérisation 1.3.24, S est minimal donc tout sommet de S appartient à un chemin de G reliant x et y et ne passant par aucun autre sommet de S. L'ajout d'un sommet universel u ajoute un chemin xuy (qui ne passe donc par aucun autre sommet de $S \cup \{u\}$) et ne change pas les autres chemins reliant x et y et passant par un unique autre sommet de $S \cup \{u\}$. Donc $S \cup \{u\}$ est un séparateur minimal du graphe obtenu en ajoutant le sommet universel u au graphe G.
◇

Lemme 4.1.21
Soit G un graphe, u un sommet universel et S un séparateur minimal de G. $G(V-\{u\})$ a notamment pour séparateur minimal $S-\{u\}$.

Preuve : Soit G un graphe, u un sommet universel et S un séparateur minimal de G. D'après la propriété 4.1.17, u appartient à S. Son retrait de G laisse donc inchangées les composantes connexes induites par S ; pareillement sont conservées les arêtes non incidentes à u et donc les chemins entre sommets de $V-S$ ne passant pas par u. Ainsi la caractérisation 1.3.24 reste valide pour $S-\{u\}$ et donc $S-\{u\}$ est un séparateur minimal

de $G(V-\{u\})$. Comme u est universel, G est connexe. Par contre le retrait de u peut déconnecter le graphe ; le raisonnement précédent s'applique à ce cas où $\emptyset$ devient séparateur minimal de $G(V-\{u\})$.
◇

On peut donc retirer tous les sommets universels d'un graphe en conservant sa structure articulatoire. L'ensemble des sommets universels d'un graphe peut être déterminé en temps linéaire ($\mathrm{O}(m)$), par un simple parcours du graphe calculant les degrés des sommets.

Rangées de zéros dans une relation

Théorème 4.1.22
*Une relation binaire $\mathcal{R}$ admet une rangée de **zéros** x si et seulement si x est un sommet universel dans le graphe sous-jacent $G_{\mathcal{R}}$.*

Preuve : Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte.

$\Longrightarrow$ Soit x une propriété correspondant à une colonne de zéros dans la table de $\mathcal{R}$. Dans le graphe sous-jacent $G_{\mathcal{R}}$, x est donc voisin de tous les objets. Puisque $\mathscr{P}$ est une clique, x est aussi voisin de toutes les propriétés. Par conséquent, x est universel. On construit dualement la preuve pour un objet i correspondant à une ligne de zéros.

$\Longleftarrow$ Soit x un sommet universel de $G_{\mathcal{R}}$. Si x est une propriété, il est ainsi voisin de tous les objets et constitue une colonne de zéros dans la table de $\mathcal{R}$. Dualement, un objet universel dans $G_{\mathcal{R}}$ correspond à une ligne de zéros de $\mathcal{R}$.

◇

La propriété 4.1.17 permet ainsi de supprimer un sommet universel dans $G_{\mathcal{R}}$ et donc sa rangée de zéros dans $\mathcal{R}$ sans modifier la structure articulatoire de $G_{\mathcal{R}}$. Par conséquent, on peut retirer les sommets universels de $G_{\mathcal{R}}$ sans altérer l'ensemble des concepts de $(\mathscr{P}, \mathscr{O}, \mathcal{R})$, puisque ces sommets n'apparaissent dans aucun élément de $\mathcal{L}(\mathcal{R})$.

Sommets simpliciaux d'un graphe

Les sommets simpliciaux sont, en quelque sorte, les opposés des sommets universels puisqu'ils respectent la propriété suivante, miroir de la propriété 4.1.17.

Propriété 4.1.23
*Un sommet est simplicial si et seulement si il n'appartient à **aucun** séparateur minimal du graphe.*

Preuve : Soit $G = (V, E)$ un graphe.

1. Soit x un sommet simplicial de G. $N(x)$ est une clique, donc on ne peut trouver deux voisins de x qui ne soient pas reliés et, par le corollaire 1.3.26, x ne peut donc pas appartenir à un séparateur minimal.

2. Soit x un sommet non simplicial de G. Son voisinage n'étant pas une clique, on peut y trouver deux sommets y et z non reliés. Puisque y et z ne sont pas voisins, le corollaire 1.3.26 nous permet d'affirmer qu'il existe un yz-séparateur minimal. Puisqu'il existe un chemin $(y, ..., x, ..., z)$ qui relie y et x en passant par x, la caractérisation 1.3.24 nous permet d'affirmer que x appartient à ce séparateur minimal.

◇

Ainsi, le retrait d'un sommet simplicial préserve l'ensemble des séparateurs minimaux d'un graphe.

Rangées de uns dans une relation

Théorème 4.1.24
*Une relation binaire $\mathcal{R}$ sans rangée de zéros admet une rangée de **uns** x si et seulement si x est un sommet simplicial dans le graphe sous-jacent $G_{\mathcal{R}}$.*

Preuve : Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte où $\mathcal{R}$ n'a pas de rangées de zéros.

$\Longrightarrow$ Soit x une propriété correspondant à une colonne de uns dans la table de $\mathcal{R}$. Dans le graphe sous-jacent $G_{\mathcal{R}}$, x n'est donc voisin d'aucun objet. Puisque $\mathscr{P}$ est une clique, le voisinage de x est une clique. Par conséquent, x est simplicial. On construit dualement la preuve pour un objet i correspondant à une ligne de uns.

$\Longleftarrow$ Soit $x \in \mathscr{P}$ un sommet simplicial de $G_{\mathcal{R}}$. $N(x) = \mathscr{P} + N^+(x)$ est donc une clique. Supposons qu'il existe $i \in N^+(x)$; puisque $N(x)$ est une clique, on a $N^+(i) = \mathscr{P}$. Ceci implique que i est une rangée de zéros dans la table de $\mathcal{R}$, ce qui est contradictoire. Par conséquent $N^+(x) = \emptyset$. Dans $G_{\mathcal{R}}$, x n'est voisin d'aucun objet, il constitue donc une rangée de uns dans la table de $\mathcal{R}$. On construit dualement la preuve pour un objet i simplicial.

◇

La propriété 4.1.23 permet ainsi de supprimer un sommet simplicial dans $G_{\mathcal{R}}$ et donc sa rangée de uns dans $\mathcal{R}$ sans modifier la structure articulatoire de $G_{\mathcal{R}}$. Par conséquent, on peut retirer les sommets simpliciaux de $G_{\mathcal{R}}$ sans altérer l'ensemble des concepts de $(\mathscr{P}, \mathscr{O}, \mathcal{R})$, puisque ces sommets apparaissent dans tous les éléments de $\mathcal{L}(\mathcal{R})$.

En conclusion, toutes les rangées de zéros et les rangées de uns peuvent être supprimées de $\mathcal{R}$ en temps linéaire $O(m)$ par un simple examen des degrés des sommets. Ces retraits ne perturbent pas la structure articulatoire de $G_{\mathcal{R}}$; ils préservent ainsi le treillis des concepts $\mathcal{L}(\mathcal{R})$: il suffira de retirer les sommets simpliciaux de l'étiquette des éléments où ils apparaissent. Nous considèrerons, par la suite, que ces rangées ont été retirées des relations.

4.2 Sélection d'un sous-treillis par saturation de séparateurs minimaux du graphe sous-jacent

4.2.1 Effet de la saturation d'un séparateur minimal du graphe sous-jacent

Dans cette sous-section, nous allons examiner ce que devient le treillis quand on sature un séparateur minimal du graphe sous-jacent. Nous utiliserons pour ce faire la notion de séparateurs minimaux croisants. Les propriétés afférentes peuvent être appliquées à notre graphe sous-jacent $G_{\mathcal{R}}$: la saturation d'un séparateur minimal S de $G_{\mathcal{R}}$ définit une nouvelle relation $\mathcal{R}'$ obtenue en retirant de $\mathcal{R}$ chaque élément (x, y) correspondant à une arête xy ajoutée à S. Grâce à la propriété 1.3.33, nous devons nous attendre à ce que tout concept de la relation finale $\mathcal{R}'$ soit un concept de la relation initiale $\mathcal{R}$. Pour le prouver, nous allons, par le lemme suivant, établir une relation entre séparateurs minimaux non croisants d'un graphe et éléments comparables d'un treillis.

Lemme 4.2.1
Soient $\mathcal{R}$ une relation binaire, $\mathcal{L}(\mathcal{R})$ le treillis des concepts associé et $G_{\mathcal{R}}$ le graphe cobiparti sous-jacent. Soient S_1 et S_2 deux séparateurs minimaux de $G_{\mathcal{R}}$ représentant respectivement les concepts $A_1 \times B_1$ et $A_2 \times B_2$. S_1 et S_2 sont deux séparateurs minimaux non croisants de $G_{\mathcal{R}}$ si et seulement si $A_1 \times B_1$ et $A_2 \times B_2$ sont deux concepts comparables de $\mathcal{L}(\mathcal{R})$.

Preuve : Soit S_1 et S_2 deux séparateurs minimaux représentant respectivement les concepts $A_1 \times B_1$ et $A_2 \times B_2$.

$\Longrightarrow$

Supposons que S_1 et S_2 sont non croisants. Par la définition 1.3.31, ceci implique (sans perte de généralité) $S_1 \cap A_2 \neq \emptyset$. Alors $A_2 \subseteq (A_1 \cup B_1)$ et (puisque $A_2 \in \mathscr{P}$ et $B_1 \in \mathscr{O}$) $A_2 \subseteq A_1$. Ainsi $A_1 \times B_1$ est un descendant de $A_2 \times B_2$; par conséquent, ces deux concepts sont comparables.

$\Longleftarrow$

Supposons que S_1 et S_2 sont croisants. Par la définition 1.3.31, $S_1 \cap A_2 \neq \emptyset$ et $S_2 \cap A_1 \neq \emptyset$. Par conséquent les concepts $A_1 \times B_1$ et $A_2 \times B_2$ ne sont pas comparables.

$\diamond$

Ce lemme nous permet de conclure :

Théorème 4.2.2
Soient $\mathcal{R}$ une relation, $\mathcal{L}(\mathcal{R})$ son treillis des concepts associé et $G_{\mathcal{R}}$ son graphe sous-jacent ; soit S un séparateur minimal de $G_{\mathcal{R}}$ représentant le concept $A \times B$ de $\mathcal{L}(\mathcal{R})$; soit $\mathcal{R}'$ la relation obtenue en saturant S. Les deux propriétés suivantes sont alors vérifiées :

1. *Le treillis des concepts $\mathcal{L}(\mathcal{R}')$ peut être obtenu en retirant de $\mathcal{L}(\mathcal{R})$ tous les éléments qui ne sont pas comparables avec $A \times B$.*
2. *Le treillis des concepts $\mathcal{L}(\mathcal{R}')$ est un sous-treillis du treillis initial $\mathcal{L}(\mathcal{R})$.*

Preuve : Soient $\mathcal{R}$ une relation, $\mathcal{L}(\mathcal{R})$ son treillis des concepts, $G_{\mathcal{R}}$ son graphe sous-jacent et S un séparateur minimal de $G_{\mathcal{R}}$. Soit $\mathcal{R}'$ la relation obtenue à partir de $\mathcal{R}$ en saturant S et soit $\mathcal{L}(\mathcal{R}')$ son treillis des concepts. Par la propriété 1.3.33, la saturation de S fait disparaître du graphe exactement les séparateurs minimaux qui ne croisent pas S. Ainsi, par le lemme 4.2.1, disparaissent de $\mathcal{L}(\mathcal{R})$ exactement les concepts qui ne sont pas comparables avec $A \times B$. En conclusion, le treillis des concepts $\mathcal{L}(\mathcal{R}')$ est un sous-treillis de $\mathcal{L}(\mathcal{R})$.
◇

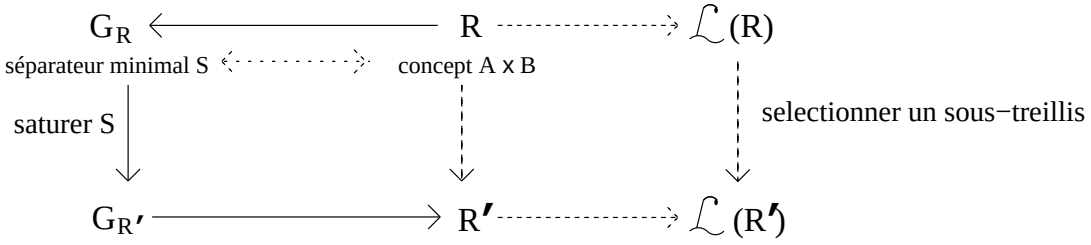


FIG. 4.4 – Rapports entre relation, graphe sous-jacent et treillis des concepts lors de la saturation d'un séparateur

.

Le théorème 4.2.2 définit un procédé qui nous permet de restreindre une relation binaire $\mathcal{R}$ à une relation $\mathcal{R}' \subset \mathcal{R}$ dont le treillis des concepts $\mathcal{L}(\mathcal{R}')$ est un sous-treillis de $\mathcal{L}(\mathcal{R})$. Ceci peut s'avérer important pour de nombreuses applications puisque le fait de restreindre arbitrairement une relation ne permet pas, en général, d'obtenir un sous-treillis et peut même aboutir à un treillis plus grand que le treillis initial.

Etape de saturation 4.2.3 d'un séparateur minimal non complet.
Donnée : un contexte ($\mathscr{P}$,$\mathscr{O}$,$\mathcal{R}$), son graphe $G_{\mathcal{R}}$ et un séparateur minimal S de $G_{\mathcal{R}}$.
Résultat : la sous-relation $\mathcal{R}'$ obtenue en saturant S dans $G_{\mathcal{R}}$.
$\mathcal{R}' \leftarrow \mathcal{R}$;
Pour toute arête xy absente de S **faire :**
Retirer (x,y) de $\mathcal{R}'$.

Exemple 4.2.4
Le séparateur $S = \{a, d, e, f, 3, 4, 5, 6\}$, dans le graphe $G_{\mathcal{R}}$ de la figure 4.3, représente le concept $bc \times 12$. Lors de sa saturation, les arêtes $a3$, $a6$, $d4$, $d5$ et $f3$ seront ajoutées, définissant une nouvelle relation $\mathcal{R}'$:

$\mathcal{R}'$	a	b	c	d	e	f
1		×	×	×	×	
2	×	×	×			
3		×				
4					×	
5			×			
6						

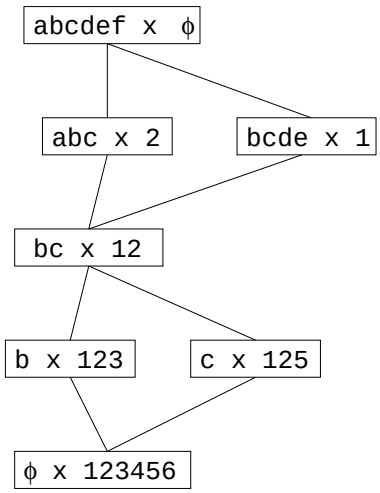


FIG. 4.5 – Le treillis $\mathcal{L}(\mathcal{R}')$ de l'exemple 4.2.4.

La figure 4.5 donne le sous-treillis $\mathcal{L}(\mathcal{R}')$ ainsi obtenu. La saturation de S a provoqué la disparition des concepts $a \times 236$, $ab \times 23$, $abf \times 3$, $d \times 145$, $cd \times 15$ et $de \times 14$ du treillis $\mathcal{L}(\mathcal{R})$.
$\diamond$

4.2.2 Equivalence entre chaînes maximales du treillis et triangulations minimales du graphe

Nous allons conclure cette section en abordant les triangulations minimales de $G_{\mathcal{R}}$. Des travaux récents ont montré que les séparateurs minimaux peuvent être utilisés pour calculer une triangulation minimale (voir le chapitre 1) :

Algorithme 4.2.5 `SATURE_SM` ([Ber98]) :
Donnée : un graphe G quelconque.
Résultat : une triangulation H de G.
Begin
Tant que il reste dans G un séparateur minimal non complet **faire :**
  Choisir un séparateur minimal S non complet de G ;
  Saturer S ;
End.

Propriété 4.2.6 *([Ber98]).*
Le schéma algorithmique 4.2.5 aboutit à une triangulation minimale d'un graphe G à n sommets en moins de n étapes de saturation.
De plus, ce procédé caractérise chaque triangulation minimale de G par son ensemble de séparateurs minimaux.

En effet, la saturation d'un séparateur minimal S d'un graphe G peut faire disparaître des séparateurs minimaux de G et peut ajouter des arêtes à d'autres séparateurs minimaux

de G, mais elle ne fait pas apparaître de nouveaux séparateurs minimaux ni ne modifie l'ensemble de sommets constituant les séparateurs minimaux restant après saturation de S.

Nous pouvons appliquer ceci au graphe sous-jacent d'une relation :

Propriété 4.2.7
Calculer une triangulation minimale d'un graphe sous-jacent $G_{\mathcal{R}}$ en saturant répétitivement un séparateur minimal non complet aboutira à un graphe d'intervalles propres $G_{\mathcal{R}''}$ et à une relation correspondante $\mathcal{R}''$ telle que $\mathcal{L}(\mathcal{R}'')$ est une chaîne maximale de $\mathcal{L}(\mathcal{R})$.

Preuve : Tout d'abord, la propriété 4.1.7 dit que toute triangulation minimale de $G_{\mathcal{R}}$ donne un graphe d'intervalles propres. Le reste de cette propriété est une conséquence directe du théorème 4.2.2. D'une part, à chaque étape de l'algorithme 4.2.5, on enlève tous les concepts non comparables avec le concept représenté par le séparateur complété à cette étape. L'algorithme s'arrête quand il ne reste plus de séparateur minimal à compléter et donc quand tous les concepts restants sont deux-à-deux comparables, formant donc une chaîne. D'autre part, en éliminant tous les concepts non comparables à un concept lors de l'étape de saturation du séparateur qui le représente, on conserve au moins dans le sous-treillis obtenu une chaîne maximale passant par ce concept.
◇

Propriété 4.2.8
L'ensemble des triangulations minimales de $G_{\mathcal{R}}$ et l'ensemble des chaînes maximales de $\mathcal{L}(\mathcal{R})$ sont en bijection.

Preuve : La propriété 4.2.7 a montré que chaque triangulation minimale de $G_{\mathcal{R}}$ correspond à une chaîne de $\mathcal{L}(\mathcal{R})$. Réciproquement, chaque chaîne maximale de $\mathcal{L}(\mathcal{R})$ peut être obtenue à partir de $\mathcal{L}(\mathcal{R})$ en supprimant tous les concepts non comparables aux éléments de cette chaîne maximale, c'est-à-dire en saturant l'ensemble des séparateurs minimaux représentant les éléments de la chaîne, grâce à une exécution de l'algorithme 4.2.5 qui aboutit à une triangulation minimale de $G_{\mathcal{R}}$. D'autre part, la propriété 4.2.6 indique qu'une triangulation H d'un graphe G est uniquement caractérisé par l'ensemble des séparateurs minimaux de H, ensemble en bijection avec les éléments de la chaîne. On a donc l'assurance d'une correspondance bijective entre les chaînes maximales de $\mathcal{L}(\mathcal{R})$ et les triangulations minimales de $G_{\mathcal{R}}$.
◇

Remarque 4.2.9 *Une chaîne maximale du treillis des concepts d'un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ a au plus $Min(|\mathscr{P}|, |\mathscr{O}|)+1$ éléments et peut être obtenue en moins de n étapes en s'appuyant sur la propriété 4.2.6. Comme, chaque fois qu'un séparateur minimal est saturé, le nombre de concepts décroît, le procédé de saturation décrit par le théorème 4.2.2 peut être répété autant de fois que nécessaire, par exemple jusqu'à aboutir à un sous-treillis de taille polynomiale. Ceci peut être très utile quand le treillis des concepts est exponentiellement grand, puisqu'on peut ainsi se limiter à l'examen d'une partie de ce treillis.*

Exemple 4.2.10
La décomposition de l'exemple 4.2.4 peut se poursuivre avec la saturation des séparateurs minimaux $\{d, e, f, 1, 3, 4, 5, 6\}$ et $\{a, c, d, e, f, 4, 5, 6\}$, obtenant ainsi une triangulation minimale de $G_{\mathcal{R}}$. La relation $\mathcal{R}''$ correspondante est :

$\mathcal{R}''$	a	b	c	d	e	f
1		×	×			
2	×	×	×			
3		×				
4						
5						
6						

Le treillis $\mathcal{L}(\mathcal{R}'')$ obtenu, donné dans la figure 4.6, est une chaîne maximale de $\mathcal{L}(\mathcal{R})$.
◇

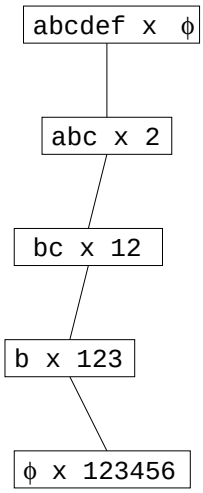


FIG. 4.6 – Le treillis $\mathcal{L}(\mathcal{R}'')$ de l'exemple 4.2.10.

4.3 Décomposition par séparateurs minimaux d'une relation et de son treillis

Dans le cas des graphes co-bipartis, le processus de décomposition par séparateurs minimaux complets d'un graphe (voir étape de décomposition 1.3.37) est considérablement simplifié du fait que chaque séparateur minimal définit seulement deux composantes connexes (caractérisation 4.1.10) : dans le graphe $G_{\mathcal{R}}$ défini par une relation binaire $\mathcal{R}$, un séparateur minimal S définit deux composantes $A \subset \mathscr{P}$ et $B \subset \mathscr{O}$. La décomposition par séparateurs minimaux complets dans un co-biparti pourrait donc se traduire par $G_1 = G_{\mathcal{R}}(A \cup S)$ et $G_2 = G_{\mathcal{R}}(B \cup S)$.

Dans ce cas, les sommets de $S \cap \mathscr{P}$ sont universels dans le graphe $G_{\mathcal{R}}(A \cup S)$ donc, par la propriété 4.1.17, ils n'apportent aucune information sur les séparateurs minimaux et peuvent être retirés du graphe. Les sommets de $S \cap \mathscr{O}$ sont pareillement universels dans

$G_{\mathcal{R}}(B \cup S)$ et peuvent être retirés. Pour un graphe co-biparti $G_{\mathcal{R}}$ provenant d'une relation binaire $\mathcal{R}$, nous allons donc définir une étape de décomposition simplifiée qui remplace $G_{\mathcal{R}}$ par $G_1 = G_{\mathcal{R}}((A \cup S) - (S \cap \mathscr{P}))$ et $G_2 = G_{\mathcal{R}}((B \cup S) - (S \cap \mathscr{O}))$. Remarquons qu'on a : $(A \cup S) - (S \cap \mathscr{P}) = A \cup (S \cap \mathscr{O}) = A \cup (\mathscr{O} - B)$.

La propriété 1.3.38 a pour conséquence que le calcul de l'ensemble des séparateurs minimaux du graphe sous-jacent initial $G_{\mathcal{R}}$ peut être réalisé séparément sur les sous-graphes plus petits définis par une étape de décomposition par séparateur minimal complet : T_1 sera un séparateur minimal de G_1 si et seulement si $T_1 \cup (S \cap \mathscr{P})$ est un séparateur minimal de $G_{\mathcal{R}}$, T_2 sera un séparateur minimal G_2 si et seulement si $T_2 \cup (S \cap \mathscr{O})$ est un séparateur minimal de $G_{\mathcal{R}}$. On peut ainsi calculer les concepts de $\mathcal{R}$ séparément sur les sous-relations définies. De plus, il est clair que G_1 contient tous les séparateurs minimaux représentant un concept qui est ancêtre de $A \times B$ et que G_2 contient tous les séparateurs minimaux représentant un concept qui est descendant de $A \times B$.

Dans un graphe co-biparti, on peut tester la présence d'un séparateur minimal complet en temps linéaire et la décomposition peut être calculée dans la même complexité (voir propriété 4.1.8).

Cependant, il n'y a en général pas forcément de séparateur minimal complet dans $G_{\mathcal{R}}$. Nous pouvons combiner ce qui précède avec les résultats de la section précédente sur la saturation et saturer un séparateur minimal non complet avant de poursuivre la décomposition du graphe.

Une des propriétés remarquables des graphes co-bipartis est que les arêtes ajoutées lors de la saturation d'un séparateur minimal ne sont recopiées dans aucun des sous-graphes définis par la précédente décomposition, puisque les arêtes ajoutées le seront entre un sommet de $S \cap \mathscr{O}$ et un sommet de $S \cap \mathscr{P}$, définissant, comme on l'a dit ci-dessus, des sommets universels. De ce fait, l'étape de décomposition sera la même que le séparateur minimal complet utilisé soit "naturel" ou "artificiel".

Nous allons donc définir les étapes de décomposition suivantes qui peuvent utiliser n'importe quel séparateur minimal qu'il soit complet ou non :

Etape de décomposition 4.3.1 d'un graphe co-biparti *:*
Soit $G_{\mathcal{R}}$ le graphe sous-jacent d'un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ et S un séparateur minimal de $G_{\mathcal{R}}$ définissant les composantes $A \subset \mathscr{P}$ et $B \subset \mathscr{O}$.
Remplacer $G_{\mathcal{R}}$ par $G_1 = G_{\mathcal{R}}(A \cup (S \cap \mathscr{O})) = G_{\mathcal{R}}(A \cup (\mathscr{O} - B))$ et $G_2 = G_{\mathcal{R}}(B \cup (S \cap \mathscr{P})) = G_{\mathcal{R}}(B \cup (\mathscr{P} - A))$.

De cette étape de décomposition de graphe, nous pouvons déduire une décomposition correspondante pour les relations.

Etape de décomposition 4.3.2 d'une relation *:*
Soient $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte, $\mathcal{L}(\mathcal{R})$ le treillis des concepts associé, $G_{\mathcal{R}}$ le graphe sous-jacent et S un séparateur minimal de $G_{\mathcal{R}}$ définissant les composantes $A \subset \mathscr{P}$ et $B \subset \mathscr{O}$. $\mathcal{R}$ peut être décomposée en deux sous-relations $\mathcal{R}_1 = \mathcal{R} \cap (A \times (\mathscr{O} - B))$ et $\mathcal{R}_2 = \mathcal{R} \cap ((\mathscr{P} - A) \times B)$ telles que :

1. *Un concept $X \times Y$ est un* ***ancêtre*** *du concept $A \times B$ dans $\mathcal{L}(\mathcal{R})$ si et seulement si $X \times (Y - B)$ est un concept de la relation $\mathcal{R}_1$. Le sous-treillis de $\mathcal{L}(\mathcal{R})$ correspondant, qui a $A \times B$ pour top, contient exactement les concepts dont l'intension incluse dans A, ce qui correspond exactement aux concepts dont l'extension est incluse dans $\mathscr{O} - B$.*
2. *Un concept $X \times Y$ est un* ***descendant*** *du concept $A \times B$ dans $\mathcal{L}(\mathcal{R})$ si et seulement si $(X - A) \times Y$ est un concept de la relation $\mathcal{R}_2$. Le sous-treillis de $\mathcal{L}(\mathcal{R})$ correspondant, qui a $A \times B$ pour bottom, contient exactement les concepts dont l'intension incluse dans $\mathscr{P} - A$, ce qui correspond exactement aux concepts dont l'extension est incluse dans B.*

On retrouve ainsi la sous-relation que définit Bordat (voir [Bor86]) pour son algorithme de génération des concepts. Notons que ce procédé permet de travailler sur des sous-relations qui peuvent être considérablement plus petites que la relation initiale.

Ce procédé nous permet de répondre efficacement à la question suivante :
« *Si on a un sous-ensemble de propriétés X (par exemple un ensemble de symptômes de patients dans une base de données médicale), dans quelle sous-relation devrait-on travailler si on veut se restreindre aux concepts qui contiennent toutes les propriétés de X ?* »
Pour répondre simplement à cette question :
– on calcule le plus petit concept dont l'intension contient X
– on extrait la sous-relation correspondant aux descendants de ce concept.

Exemple 4.3.3
Utilisons le séparateur minimal $S = \{a, d, e, f, 3, 4, 5, 6\}$ de l'exemple 4.1.2 (page 89). S définit les composantes $C_1 = \{b, c\}$ et $C_2 = \{1, 2\}$, représentant ainsi le concept $bc \times 12$. $S \cap \mathscr{O} = \{3, 4, 5, 6\}$ et $S \cap \mathscr{P} = \{a, d, e, f\}$.

Une étape de décomposition par S fournira $G_1 = G_{\mathcal{R}}(C_1 \cup (S \cap \mathscr{O})) = G_{\mathcal{R}'}(\{b, c, 3, 4, 5, 6\})$ et $G_2 = G_{\mathcal{R}}(C_2 \cup (S \cap \mathscr{P})) = G_{\mathcal{R}'}(\{a, d, e, f, 1, 2\})$, comme le montre la figure 4.7 (dans cette figure, les arêtes internes aux cliques $\mathscr{P}$ et $\mathscr{O}$ ne sont pas représentées).

La relation initiale $\mathcal{R}$ de l'exemple 4.1.2 et ses sous-relations R_1 et R_2 obtenues par décomposition sont données dans la figure 4.8.

Un parcours en temps linéaire de G_1 montrera clairement que les sommets 4 et 6 sont devenus universels et peuvent être retirés. La figure 4.9 montre le graphe très réduit $G_{\mathcal{R}'1}$ finalement obtenu. Les séparateurs minimaux de $G_{\mathcal{R}'1}$ sont $\{b, 3\}$ et $\{c, 5\}$, ils correspondent aux concepts $b \times 3$ et $c \times 5$. Dans le graphe entier, la réintégration de la composante $C_2 = \{1, 2\}$ fournira sans coût supplémentaire les concepts du treillis originel $b \times 123$ et $c \times 125$. Ces concepts sont précisément les prédécesseurs de $bc \times 12$.

Dans G_2, le sommet f est devenu universel ; un parcours en temps linéaire montrera que les sommets d et e partagent maintenant le même voisinage, ils peuvent être contractés sans perte d'information sur les séparateurs minimaux du graphe.

Le graphe $G_{\mathcal{R}'2}$ qui en résulte, montré dans la figure 4.9, est aussi réduit à quatre sommets. Ses séparateurs minimaux représentent $a \times 2$ et $de \times 1$, ce qui, après réintégration de $C_1 = \{b, c\}$, définit les concepts $abc \times 2$ et $bcde \times 1$, lesquels sont les successeurs de $bc \times 12$.

La décomposition de treillis correspondante est illustrée par la figure 4.10.

◇

4.4 Comment assurer que le treillis d'une relation soit de taille polynomiale

La taille exponentielle des treillis de concepts est un inconvénient majeur. Cette taille rend difficile, voire impossible, la génération du treillis. Une solution possible est de définir une sous-relation qui préserve les informations essentielles tout en admettant un nombre polynomial de concepts. Dans le cadre notamment des échantillonnages à des fins d'apprentissage, il peut être intéressant de choisir un échantillon qui, tout en étant représentatif, ne définit de même qu'un nombre polynomial de concepts.

On sait que plus la proportion de uns dans une relation est importante plus il y aura de concepts dans le treillis associé. Pour diminuer la taille du treillis et le rendre mieux gérable, on peut donc essayer de supprimer des croix de la relation. Par ce procédé, malheureusement, on n'obtient généralement pas un sous-treillis du treillis initial ; il y a même des cas où ce retrait augmente au contraire le nombre de concepts définis par la relation. Plus grave encore, en dehors des cas triviaux (relation complète, relation vide, diagonale de uns, diagonale de zéros), on ne sait pas dans quelles conditions on va pouvoir faire diminuer le nombre de concepts ou définir une relation possédant un nombre gérable de concepts.

En application directe de ce qui a été présenté dans la première section, nous sommes en mesure de proposer une approche formelle à la définition de classes de relations possédant un nombre polynomial de concepts.

En effet, en vertu du théorème 4.1.11, nous savons que le nombre de concepts définis par une relation est égal (au deux éléments $\perp$ et $\top$ près) au nombre de séparateurs minimaux du graphe sous-jacent. En choisissant une relation dont le graphe sous-jacent appartient à une classe de graphes ayant un nombre polynomial de séparateurs minimaux, nous assurons au treillis associé une taille polynomiale.

Plusieurs classes de graphes sont réputées n'admettre qu'un nombre polynomial de séparateurs minimaux. La classe des graphes triangulés, par exemple n'admet que $O(n)$ séparateurs minimaux. Nous avons vu cependant, par la propriété 4.2.8, que les treillis des concepts correspondants sont réduits à une chaîne. Il faut donc chercher d'autres classes de graphes dotées d'un nombre polynomial de séparateurs minimaux ainsi que de propriétés sur ces séparateurs. La classe la plus intéressantes, à première vue, est celle des graphes faiblement triangulés, avec seulement $O(m)$ séparateurs minimaux (voir [BBH00]) et des propriétés très semblables à celles des triangulés, en particulier en ce qui concerne les séparateurs (voir [BB00]).

Un autre aspect intéressant des graphes faiblement triangulés est que, comme nous l'avons dit au chapitre 1, ils sont pourvus d'algorithmes de reconnaissance efficaces ainsi que de schémas caractérisants de composition par ajout d'arêtes, ce qui rend facile le maintien d'une relation binaire modifiée dont le graphe sous-jacent est faiblement trian-

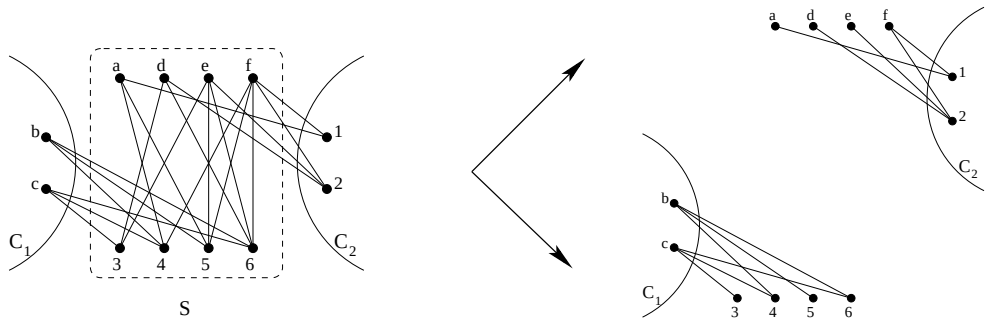


FIG. 4.7 – Graphes G_1 et G_2 issus de la décomposition de l'exemple 4.3.3 (les arêtes internes de $\mathscr{P}$ et $\mathscr{O}$ ne sont pas représentées).

	a	b	c	d	e	f
1		x	x	x	x	
2	x	x	x			
3	x	x				x
4				x	x	
5			x	x		
6	x					

	b	c
3	x	
4		
5		x
6		

	a	d	e	f
1		x	x	
2	x			

FIG. 4.8 – Relations R_1 et R_2 issues de la décomposition de l'exemple 4.3.3.

FIG. 4.9 – Les graphes $G_{\mathcal{R}'1}$ et $G_{\mathcal{R}'2}$ très réduits, issus de l'étape de décomposition de $G_{\mathcal{R}}$ par séparateur minimal de l'exemple 4.3.3.

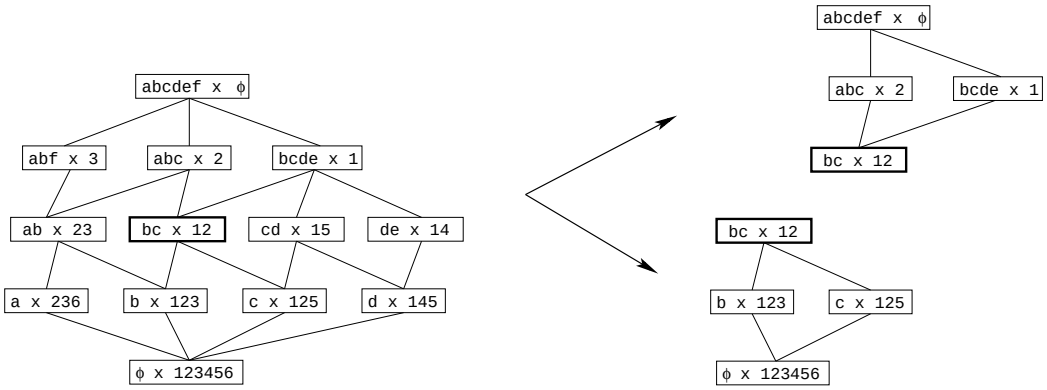


FIG. 4.10 – Treillis issus de l'étape de décomposition de l'exemple 4.3.3.

gulé.

Pour plonger le graphe sous-jacent $G_{\mathcal{R}}$ dans un graphe faiblement triangulé, nous utilisons un schéma de retrait d'arêtes qui a été défini par [RTL76] dans le cadre du calcul d'une triangulation minimale. Nous étendons ce procédé aux graphes faiblement triangulés :

Procédé algorithmique 4.4.1

Données : un graphe $G_{\mathcal{R}} = (V, E)$.
Résultat : un plongement de $G_{\mathcal{R}}$ dans un graphe faiblement triangulé $H = (V, F)$.
$F \leftarrow E$;
Tant que H n'est pas un graphe critique **faire :**
  Choisir une arête α dans $F - E$;
  Retirer α de H ;

Par *graphe critique* nous entendons ici un graphe faiblement triangulé pour lequel le retrait d'une arête donne un graphe qui n'est plus faiblement triangulé ; par analogie avec les triangulés nous pouvons définir ce premier graphe comme une **faible triangulation** du graphe $G_{\mathcal{R}}$. Notons bien que, contrairement à ce qui se passe pour les triangulés, nous ne pouvons pas assurer que ce procédé fournisse une faible triangulation qui soit **minimale**, dans le même sens que pour les triangulations minimales, bien que nous conjecturions que cela soit le cas. Cette conjecture intéresse la communauté des chercheurs spécialisés dans les faiblement triangulés depuis plus d'un an, mais résiste encore et semble constituer un problème difficile.

Nos premières expérimentations montrent qu'on aboutit assez souvent à un sous-treillis du treillis de départ. Si les arêtes ajoutées au graphe $G_{\mathcal{R}}$ ne saturent pas de séparateur minimal, ce qui est le cas général, le procédé algorithmique 4.4.1 évite de créer des points d'articulations dans le treillis. Lorsqu'on n'obtient pas de sous-treillis, on obtient un treillis qui est structurellement "proche".

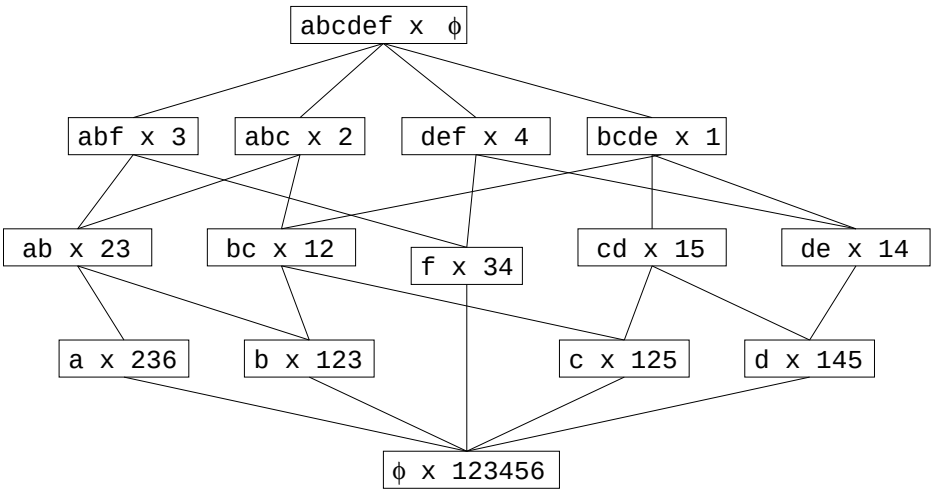


FIG. 4.11 – Le treillis des concepts $\mathcal{L}(\mathcal{R})$ de la relation $\mathcal{R}$ de l'exemple 4.4.2.

Exemple 4.4.2
Prenons le contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$, avec $\mathscr{P}= \{a, b, c, d, e, f\}$, $\mathscr{O}= \{1, 2, 3, 4, 5, 6\}$, et la relation $\mathcal{R}$ définie par la table suivante :

	a	b	c	d	e	f
1		×	×	×	×	
2	×	×	×			
3	×	×				×
4				×	×	×
5			×	×		
6	×					

Le treillis des concepts associé $\mathcal{L}(\mathcal{R})$ est montré dans la figure 4.11.
Le graphe sous-jacent $G_{\mathcal{R}}$ est montré dans la figure 4.12.
◇

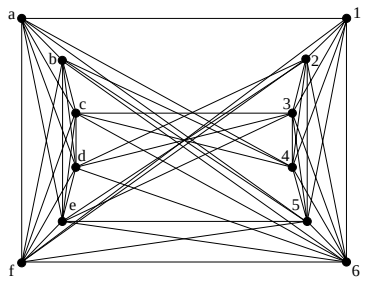


FIG. 4.12 – Le graphe $G_{\mathcal{R}}$ sous-jacent au contexte de l'exemple 4.4.2.

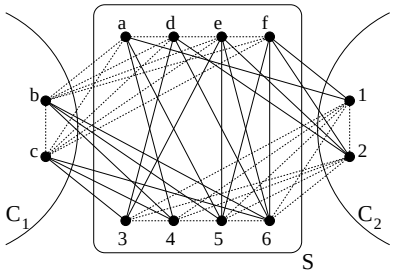


FIG. 4.13 – Le séparateur S de $G_{\mathcal{R}}$ de l'exemple 4.4.3.

Exemple 4.4.3
Dans l'exemple 4.4.2, $S = \{a, d, e, f, 3, 4, 5, 6\}$ est un séparateur minimal de $G_{\mathcal{R}}$, il sépare $C_1 = \{b, c\}$ de $C_2 = \{1, 2\}$ (ce qui est illustré par la figure 4.13) et représente le concept $bc \times 12$. On remarque qu'on a $N(b) = \{a, c, d, e, f, 4, 5, 6\}$, $N(c) = \{a, b, d, e, f, 3, 4, 6\}$ et donc bien $N(C_1) = (N(b) \cup N(c)) - C_1 = \{a, d, e, f, 3, 4, 5, 6\} = S$.

Lors de la saturation du séparateur minimal S, les arêtes $a3$, $a6$, $d4$, $d5$, $e4$, $f3$ et $f4$ seront ajoutées à $G_{\mathcal{R}}$ et les croix correspondantes retirées de la relation, définissant ainsi une nouvelle relation $\mathcal{R}'$:

$\mathcal{R}'$	a	b	c	d	e	f
1		×	×	×	×	
2	×	×	×			
3		×				
4						
5			×			
6						

La figure 4.14 donne le treillis $\mathcal{L}(\mathcal{R}')$ obtenu, de taille bien plus réduite. La saturation de S a provoqué la disparition des concepts $a \times 236$, $ab \times 23$, $abf \times 3$, $d \times 145$, $cd \times 15$; $f \times 34$, $def \times 4$ et $de \times 14$ du treillis.
◇

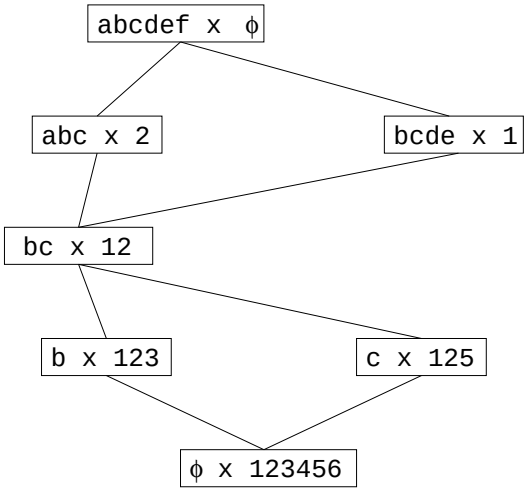


FIG. 4.14 – Le treillis des concepts $\mathcal{L}(\mathcal{R}')$ de la relation $\mathcal{R}'$ de l'exemple 4.4.3.

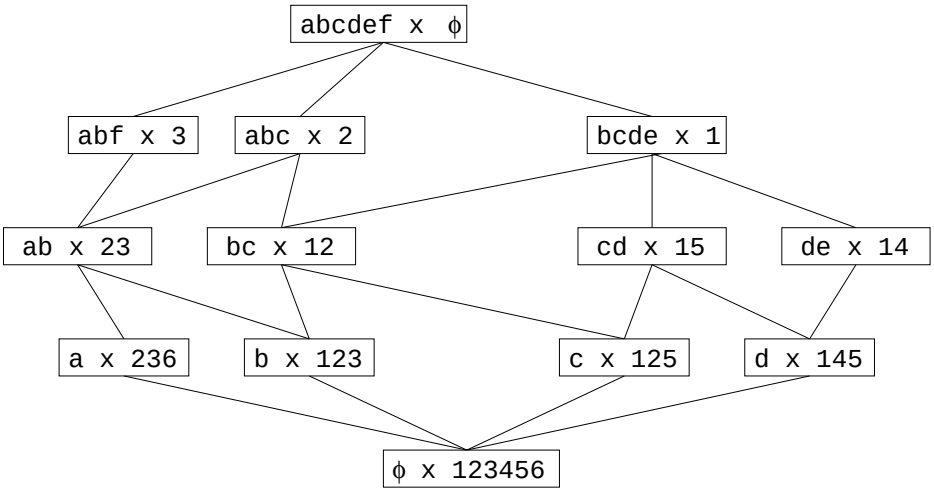


FIG. 4.15 – Le treillis des concepts $\mathcal{L}(\mathcal{R}'')$ de la relation $\mathcal{R}''$ de l'exemple 4.4.4.

Exemple 4.4.4

Dans l'exemple 4.4.2, le graphe $G_{\mathcal{R}}$ de la relation initiale $\mathcal{R}$ n'est pas faiblement triangulé. L'ajout de l'arête $f4$ à $G_{\mathcal{R}}$ plongera le graphe dans un graphe faiblement triangulé. On obtient une nouvelle relation $\mathcal{R}''$, sans la croix $(f, 4)$:

	a	b	c	d	e	f
1		×	×	×	×	
2	×	×	×			
3	×	×				×
4				×	×	
5			×	×		
6	×					

Le treillis $\mathcal{L}(\mathcal{R}'')$ correspondant est montré dans la figure 4.15. On remarque qu'il est plus intéressant que le treillis $\mathcal{L}(\mathcal{R}')$ obtenu par saturation d'un séparateur minimal (figure 4.14). Cette technique permet de choisir des treillis de petite taille, c'est ce que nous avons fait avec le treillis de la figure 4.2 (page 90), identique au treillis $\mathcal{L}(\mathcal{R}'')$ de cet exemple.

$\diamond$

Ce que que nous avons présenté dans cette section, constitue une application très utile de la classe des graphes faiblement triangulés. Nous pouvons espérer un regain d'intérêt pour cette classe qui permettra peut-être de résoudre rapidement les questions théoriques difficiles que nous laissons ouvertes.

Chapitre 5

La domination : une notion clef

Nous avons vu au chapitre précédent que la notion de séparateur minimal était fondamentale, non seulement comme outil général de manipulation des graphes, mais aussi pour l'étude des treillis. Cependant, il faut lui adjoindre une deuxième notion qui, elle, a été peu étudiée jusqu'à présent mais qui, pour les co-bipartis, s'avère tout aussi importante : la domination entre sommets d'un graphe. Intuitivement, un treillis des concepts diffère d'un treillis des parties dans la mesure où il apparaît des dominations dans les graphes sous-jacents.

Dans ce chapitre, nous présentons tout d'abord la notion de domination, puis établissons quelques propriétés afférentes. Cela nous amènera à détailler cette notion, puis à définir un ordre partiel de domination sur des ensembles de sommets. Dans la troisième section, nous étudions comment intervient cette domination quand on passe d'un élément du treillis à un autre. Dans la quatrième section, enfin, nous proposons une structure de données et des algorithmes permettant de gérer efficacement la mise à jour de cet ordre de domination au gré des modifications de la relation. Nous nous appuierons sur ce travail dans le chapitre suivant.

5.1 La domination classique

La domination a été introduite il y a une dizaine d'années sous la forme d'ensemble dominant :

Définition 5.1.1 Ensemble dominant *([BLS99]).*
Soit $G = (V, E)$ un graphe et $S \subseteq V$. S est appelé ***ensemble dominant*** *quand chaque sommet extérieur à S a un voisin dans S, c'est-à-dire $N(S) \cup S = V$.*

Les ensembles dominants permettent de caractériser les graphes parfaits :

Propriété 5.1.2 *([DHMO94], [HM93]).*
Un graphe G est parfait si et seulement si tout sous-graphe H de G a un ensemble dominant qui intersecte toutes les cliques maximales de H.

De la notion d'ensemble dominant découlent les notions de paire dominante et de sommet dominant :

Définition 5.1.3 PAIRE DOMINANTE *([COS95]).*
Deux sommets x et y d'un graphe G forment une ***paire dominante*** *quand tout chemin reliant x à y est un ensemble dominant.*

Cette notion s'avère très importante pour l'étude des graphes AT-free, puisque [COS95] a montré que ces graphes possèdent toujours une paire dominante, que l'algorithme `Lex-BFS` peut trouver en temps linéaire ([COS97]).

La domination entre sommets a été étudiée par Dahlhaus, Hammer, Maffrey et Olariu (voir [DHMO94], voir aussi [BLS99]) pour étendre la classe des graphes triangulés :

Définition 5.1.4 SOMMET DOMINANT.
Soit $G = (V, E)$ un graphe, x et y deux sommets de G. On dit que x ***domine*** *y dans G quand $N(y) \subseteq N(x) \cup \{x\}$. Un sommet x est dit* ***dominant*** *(*resp. ***dominé****) dans G quand il existe un sommet y qu'il domine (*reps. *qui le domine) ; il est dit* ***non-dominant*** *(resp.* ***non-dominé****) sinon.*

Autrement dit, x domine y quand tout voisin de y est aussi voisin de x. Par cette définition, les sommets x et y considérés pour la relation de domination ne sont pas forcément voisins.

Cette notion a permis à [DHMO94] d'étendre la notion de schéma délimination simplicial :

Définition 5.1.5 D.E.O., GRAPHE DE DOMINATION *([DHMO94]).*
Un ordre d'élimination par domination (domination elimination ordering, DEO) sur un graphe $G = (V, E)$ est un ordre total $(v_1, ..., v_n)$ sur les sommets de G tel que tout sommet v_i, $i \in [1..n-1]$ a dans $G(V - \{v_{i+1}, ..., v_n\}$ un sommet qui le domine.
Un graphe est appelé ***graphe de domination*** *si tous ses sous-graphes induits ont un DEO.*

Dans ce papier, les auteurs font la différence entre domination forte et faible :

Définition 5.1.6 DOMINATION FORTE, FAIBLE
Soit G un graphe, soient x et y deux sommets de G tels que x domine y.
- *On dit que la domination est* ***forte*** *si x et y sont voisins ;*
- *On dit que la domination est* ***faible*** *si x et y ne sont pas voisins.*

La définition 5.1.5 ne considère pas qu'un sommet puisse se dominer lui-même. Cependant, pour la suite de notre propos, en particulier dans les définitions de préordres et d'ordres, nous aurons besoin de discuter de la réflexivité de la relation de domination.

Nous détaillerons pour les graphes sans boucles, ce qui est la convention généralement adoptée, quatre situations qui sont résumées dans le tableau ci-dessous.

x domine y dans G	domination (x et y pas forcément voisins)	domination **forte** (x et y voisins)
domination **large** (x domine x)	$N(y) \subseteq N(x) \cup \{x\}$ (définition 5.1.4)	$N(y) \cup \{y\} \subseteq N(x) \cup \{x\}$
domination **stricte** (x ne domine pas x)	$N(y) \subseteq N(x) \cup \{x\}$ et $x \neq y$	$N(y) \cup \{y\} \subseteq N(x)$

Exemple 5.1.7
Domination dans le graphe G de l'exemple 1.3.7 ; ce graphe est rappelé dans la figure 5.1. Dans G, $N(a) = \{c, d, i\}$ et $N(b) = \{c, d, i\}$; par conséquent, b domine a et réciproquement. Cette domination est faible car a et b ne sont pas voisins.
$N(f) = \{e, g, h\}$ et $N(g) = \{e, f, h\}$; par conséquent f domine g et réciproquement. Cette domination est forte car f et g sont voisins.
$N(c) = \{a, b, d, h, i\}$, $N(i) = \{a, b, c, d\}$ et $N(a) = \{c, d, i\}$; par conséquent, c domine i qui domine a. Ces deux dominations sont fortes car i est voisin de a et de c.
◇

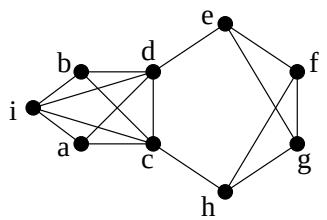


Fig. 5.1 – Le graphe G de l'exemple 5.1.7.

5.2 Propriétés remarquables de la domination

5.2.1 Domination dans un graphe quelconque

Telle qu'elle est a été présentée dans la définition 5.1.4, la domination possède un certain nombre de propriétés intéressantes.

Propriété 5.2.1
Dans un graphe, un sommet universel domine tous les autres sommets.

Preuve : Soit $G = (V, E)$ un graphe ; soient x un sommet universel et $y \in V - \{x\}$.
$N(x) = V - \{x\}$, puisque x est universel et $N(y) \subseteq V$ donc $N(y) \subseteq N(x) \cup \{x\}$. Par conséquent x domine y.
◇

Remarquons que cette domination est forte.

Propriété 5.2.2
Dans un graphe, un sommet simplicial est dominé par chaque sommet de son voisinage.

Preuve : Soit G un graphe, soient x un sommet simplicial et $y \in N(x)$. Puisque x est simplicial, $N(x)$ est une clique. Ainsi, tout voisin de x est voisin de y. Par conséquent, x domine y.
$\diamond$

Remarquons ici aussi que cette domination est forte.

Théorème 5.2.3
Dans un graphe, deux sommets voisins x et y appartiennent à un même maxmod si et seulement si x domine y et y domine x.

Preuve : Soit $G = (V, E)$ un graphe, soient x et y deux sommets distincts de G.

$\Longrightarrow$

Si x et y appartiennent à un même maxmod X, alors $N(x) - X = N(y) - X$. Comme X est complet, on a $X \subseteq N(x) \cup \{x\}$ et $X \subseteq N(y) \cup \{y\}$ et donc $N(x) \cup \{x\} = N(y) \cup \{y\}$. Donc $N(y) \subseteq N(x) \cup \{x\}$ et $N(x) \subseteq N(y) \cup \{y\}$, par conséquent x domine y et réciproquement.

$\Longleftarrow$

Si x domine y et y domine x, on a $N(y) \subseteq N(x) \cup \{x\}$ et $N(x) \subseteq N(y) \cup \{y\}$. On a donc $N(x) - \{y\} = N(y) - \{x\}$. x et y appartiennent donc à un même module $\{x, y\}$ qui est complet puisque x et y sont voisins. Si ce module complet n'est pas maximal, on ajoute autant de voisins z de x et y que nécessaire pour qu'il devienne maximal. Pour chaque voisin z ajouté dans le module, z dominera x et y et réciproquement, comme l'a montré la première partie de la preuve.

$\diamond$

Propriété 5.2.4
La relation de domination établie sur un graphe est transitive.

Preuve : Soit $G = (V, E)$ un graphe, soient x, y et z trois sommets de G tels que x domine y et y domine z.
On a donc $N(y) \subseteq N(x) \cup \{x\}$ et $N(z) \subseteq N(y) \cup \{y\}$. Par conséquent, $N(z) \subseteq (N(x) \cup \{x\}) \cup \{y\}$. Si y et z sont voisins, alors x et z le sont aussi (puisque x domine y), et alors x et y sont aussi voisins (puisque y domine z), donc $y \in N(x)$ et donc $N(z) \subseteq N(x) \cup \{x\}$; sinon, $y \notin N(z)$ et on a aussi $N(z) \subseteq N(x) \cup \{x\}$. Dans les deux cas, x domine z.
$\diamond$

Remarque 5.2.5 *La transitivité reste valide si on se restreint à la domination forte, puisque dans le cas où les trois sommets x, y et z considérés dans la preuve sont tous voisins a été examiné.*

Le théorème 5.2.3 apporte un nouvel éclairage sur la relation d'équivalence établie entre les sommets d'un maxmod par le théorème 1.3.6.

Pour définir l'ordre de domination, nous nous appuierons sur la **domination large et forte** qui constitue un préordre sur les sommets d'un graphe. La relation d'équivalence partitionnant les sommets du graphe en maxmods permet d'obtenir, par contraction de ces maxmods, l'ordre quotient canonique du préordre. Il est indispensable de choisir une domination forte pour pouvoir construire les maxmods, puisque ce sont des ensembles de sommets voisins.

Nous verrons par la suite que, dans un graphe co-biparti, toutes les dominations sont des dominations fortes.

Théorème 5.2.6
Soit G un graphe et H le graphe quotient obtenu en contractant tous les maxmods dans G. La relation de domination forte sur H est un ordre (ordre strict ou large selon que la relation de domination choisie est stricte ou large). Cet ordre sera appelé ***ordre de domination*** *de G.*
Nous noterons $\tilde{x}$ le maxmod auquel x appartient.

Preuve : Soit G un graphe et H le graphe obtenu par contraction des maxmods dans G.
Dans H, les maxmods sont réduits à des sommets. Le théorème 5.2.3 nous permet d'affirmer que, dans H, la relation de domination forte ne contient aucun arc de symétrie. Nous savons déjà que cette relation est un préordre (réflexive par convention, transitive par la propriété 5.2.4), elle forme donc un ordre. En supprimant les boucles de réflexivité, nous obtenons l'ordre strict associé.
◇

Exemple 5.2.7
Dans le graphe G de l'exemple 5.1.7 (figure 5.1, page 115) :
a est un sommet simplicial, il est donc dominé par chaque sommet de son voisinage $N(a) = \{c, d, i\}$. a et b forment un module, mais ce module n'est pas complet puisque ces deux sommets ne sont pas reliés ; on ne peut donc pas appliquer le théorème 5.2.3 à cette paire de sommets. Si on n'examine la domination forte, a ne domine pas b et b ne domine pas a.
c domine i qui domine a ; par transitivité, c domine a ; toutes ces dominations sont fortes puisque aci sont deux à deux voisins.
Les deux sommets voisins f et g forment un maxmod, ils se dominent l'un l'autre. Il n'y a pas d'autre maxmod non trivial dans G.
Le graphe quotient de G obtenu en contractant $\{f, g\}$ est présenté dans la figure 5.2. L'ordre de domination correspondant est présenté dans la figure 5.2.
◇

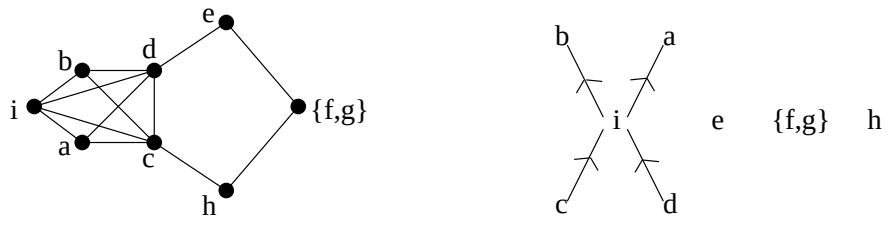


FIG. 5.2 – Le graphe quotient et l'ordre de domination associés au graphe G de l'exemple 5.2.7.

5.2.2 Domination dans le graphe sous-jacent à une relation

Nous allons maintenant étudier les rapports de domination qui existent entre les sommets d'un graphe associé à une relation. Ce graphe étant co-biparti, cela va induire un certain nombre de propriétés intéressantes. Ces propriétés concerneront des graphes dépourvus de sommet universel : la propriété 5.2.1 a établi que ce type de sommet domine tous les sommets du graphe, et nous avons montré, dans le chapitre précédent, qu'un sommet universel du graphe associé à une relation correspond à une ligne de zéros de cette relation.

Propriété 5.2.8
Soit $G = (X+Y, E)$ un graphe co-biparti sans sommet universel, construit sur les cliques X et Y. Un élément de X ne peut dominer un élément de Y, ni un élément de Y dominer un élément de X.

Preuve : Soit $G = (V, E)$ un graphe co-biparti composé à partir des cliques X et Y, $V = X+Y$.
On va démontrer que si un sommet de X domine un sommet de Y alors G a un sommet universel. Supposons que $x \in X$ domine $y \in Y$ dans G. On a donc $N(y) \subseteq N(x) \cup \{x\}$. Comme Y est une clique, $Y \subseteq N(y) \cup \{y\}$. Par conséquent $Y \subseteq N(x) \cup \{x, y\}$ et donc x est universel dans G.
On démontre dualement que si un élément de Y domine un élément de X alors G a un sommet universel.
◇

Par cette propriété, toute domination dans un graphe co-biparti est une domination forte, ne concernant que deux sommets voisins.

Propriété 5.2.9
Dans un graphe co-biparti sans sommet universel, deux sommets x et y appartiennent à un même maxmod si et seulement si x domine y et y domine x.

Preuve : Cette propriété se déduit trivialement du théorème 5.2.3 appliqué à un graphe co-biparti sans sommet universel, dans lequel la domination ne peut relier que des sommets voisins (propriété 5.2.8).
◇

Ces deux propriétés s'appliquent au graphe sous-jacent à une relation :

Théorème 5.2.10
Soit $G_{\mathcal{R}}$ le graphe sous-jacent à une relation $\mathcal{R}$ sans rangée de zéros.
- *Deux propriétés (*resp. *objets) x et y appartiennent à un même maxmod si et seulement si x domine y et y domine x.*
- *Une propriété ne peut dominer un objet, ni un objet dominer une propriété.*

Ainsi, dans le graphe sous-jacent à une relation, nous distinguerons une domination entre objets et une domination entre propriétés. L'ordre de domination sur les maxmods défini grâce au théorème 5.2.6 sera donc composé de deux ordres indépendants : un **ordre de domination sur les objets** et un **ordre de domination sur les propriétés**.

Comme toutes les dominations considérées seront fortes, il n'est pas nécessaire de considérer les arêtes internes à $G_{\mathcal{R}}$. En utilisant la notation $N^+(x)$ définie dans le chapitre 4, nous pouvons reformuler le théorème 5.2.10 comme suit :

Théorème 5.2.11
Soit $G_{\mathcal{R}}$ le graphe sous-jacent à une relation $\mathcal{R}$ sans rangée de zéros.
- *Deux sommets x_1 et x_2 de $G_{\mathcal{R}}$ appartiennent à un même maxmod si et seulement si $N^+(x_2) = N^+(x_1)$.*
- *$\tilde{x}$ domine $\tilde{y}$ dans $G_{\mathcal{R}}$ si et seulement si $N^+(y) \subset N^+(x)$.*

Preuve : Soient $G_{\mathcal{R}}$ le graphe sous-jacent à un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ et $x, y \in \mathscr{P}$.

• Nous allons d'abord démontrer que x domine y dans $G_{\mathcal{R}}$ si et seulement si $N^+(y) \subset N^+(x)$.

$\Longrightarrow$

Si x domine y, alors $N(y) \subseteq N(x) \cup (x)$. Comme $\mathscr{P} \subseteq N(y)$ et $\mathscr{P} \subseteq N(x)$, on a $(N(y) - \mathscr{P}) \subseteq ((N(x) \cup \{x\}) - \mathscr{P})$; autrement dit $N^+(y) \subseteq N^+(x)$.

$\Longleftarrow$

Réciproquement, supposons qu'on ait $N^+(y) \subseteq N^+(x)$. Comme $\mathscr{P}$ est une clique, $\mathscr{P} \subseteq N(y)$ et $\mathscr{P} \subseteq N(x)$, par conséquent $N(y) \subseteq N(x) \cup \{x\}$. x domine donc y.

Comme tous les sommets d'un même maxmod ont le même comportement pour la domination, on en déduit que $\tilde{x}$ domine $\tilde{y}$ dans $G_{\mathcal{R}}$ si et seulement si $N^+(y) \subset N^+(x)$

• Nous allons maintenant démontrer que $\tilde{x}$ domine $\tilde{y}$ dans $G_{\mathcal{R}}$ si et seulement si $N^+(y) \subset N^+(x)$.

Par le théorème 5.2.10, x et y appartiennent à un même maxmod si et seulement si x domine y et réciproquement ; par ce qui précède, on a alors $N^+(y) \subseteq N^+(x)$ et $N^+(x) \subseteq N^+(y)$ et donc $N^+(y) = N^+(x)$.

La preuve sur des objets i et j est construite semblablement.

$\diamond$

Exemple 5.2.12
Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un concept avec $\mathscr{P} = \{a, b, c, d, e, f\}$, $\mathscr{O} = \{1, 2, 3, 4, 5, 6\}$ et la $\mathcal{R}$ définie par la table suivante :

$\mathcal{R}$	a	b	c	d	e	f	g	h
1		×	×	×	×			
2	×	×	×				×	×
3	×	×				×	×	×
4				×	×			
5			×	×				
6	×							×

Le graphe sous-jacent $G_{\mathcal{R}}$ est présenté dans la figure 5.3.

On a :
$N^+(a) = \{1,4,5\}$, $N^+(b) = \{4,5,6\}$, $N^+(c) = \{3,4,6\}$, $N^+(d) = \{2,3,6\}$, $N^+(e) = \{2,3,5,6\}$, $N^+(f) = \{1,2,4,5,6\}$, $N^+(g) = \{1,4,5,6\}$, $N^+(h) = N^+(a)$ et $N^+(1) = \{a,f,g,h\}$, $N^+(2) = \{d,e,f\}$, $N^+(3) = \{c,d,e\}$, $N^+(4) = \{a,b,c,f,g,h\}$, $N^+(5) = \{a,b,e,f,g,h\}$, $N^+(6) = \{b,c,d,e,f,g\}$.

Pour les propriétés :
a et h partagent le même voisinage extérieur, ils forment un maxmod ; on pourra donc les contracter en un unique sommet $\{a,h\}$. f domine g, g domine a et b ; par transitivité, f domine aussi a et b. c n'est ni dominé ni dominant. e domine d.
Pour les objets :
1 est dominé par 4 et 5. 6 domine 2 et 3.

L'ordre de domination correspondant est présenté dans la figure 5.4.
◇

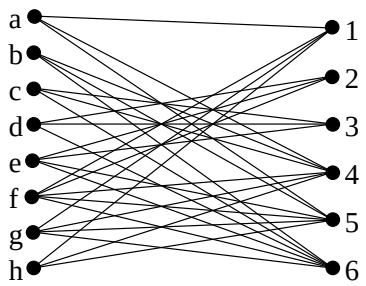


FIG. 5.3 – Le graphe $G_{\mathcal{R}}$ de l'exemple 5.2.12.

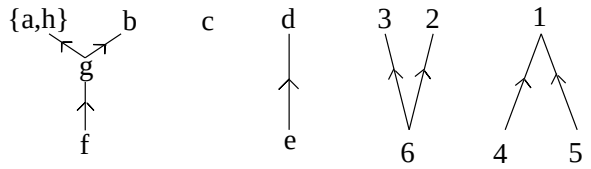


FIG. 5.4 – L'ordre de domination sur les sommets du graphe $G_{\mathcal{R}}$ de la figure 5.3.

5.3 Moplex et atomicité

La domination dans le graphe sous-jacent à une relation permet de compléter ce qui a été dit au chapitre 4 sur la structure articulatoire d'un graphe (page 95 et suivantes) par l'examen de ses maxmods.

Nous nous intéressons ici d'abord aux maxmods particuliers que sont les moplex afin d'établir une équivalence entre éléments atomiques d'un treillis et moplex du graphe sous-jacent. Nous en déduisons une équivalence entre maxmods non-dominants de $G_{\mathcal{R}}$ et éléments atomiques de $\mathcal{L}(\mathcal{R})$. Ceci permet alors de caractériser les concepts qui forment la couverture d'un élément de $\mathcal{L}(\mathcal{R})$.

5.3.1 Maxmods et Moplex dans un graphe quelconque

La propriété 1.3.6 permet de contracter un maxmod X en le remplaçant par un unique sommet représentatif de degré $|N(X)|$, sans pour autant modifier l'ensemble des séparateurs minimaux du graphe. La contraction d'un maxmod en un unique sommet représentatif peut aboutir à un sommet simplicial dont le retrait, comme indiqué précédemment, complète alors la simplification. Ainsi, le retrait d'un **maxmod simplicial** $\tilde{x}$ préserve l'ensemble des séparateurs minimaux du graphe, à l'exception de $N(\tilde{x})$.

L'algorithme de Hsu et Ma (voir [HsM99]) calcule l'ensemble des maxmods d'un graphe quelconque en temps linéaire par un procédé d'affinement de partitions.

Les moplex sont des maxmods particuliers. Définis initialement dans ([BB98]) par Berry et Bordat pour généraliser la notion de sommet simplicial et formaliser la notion d'extrémité dans un graphe, les moplex ont été utilisés dans [BB01] pour étendre la notion de triplet astéroïdal de sommets à celle de triplet astéroïdal de moplex. Par définition, les moplex apparaissent dans le voisinage de séparateurs.

Définition 5.3.1 Moplex *([BB98]).*
*Un **moplex** est un maxmod dont le voisinage est un séparateur minimal.*

Propriété 5.3.2 *([BB98]).*
Dans un graphe, la contraction d'un moplex ne modifie pas l'ensemble des séparateurs minimaux du graphe.

5.3.2 Equivalence entre atomicité dans un treillis et moplex du graphe sous-jacent

Les moplex du graphe sous-jacent $G_{\mathcal{R}}$ correspondent exactement aux extrémités non triviales du treillis $\mathcal{L}(\mathcal{R})$: ses atomes et co-atomes.

Propriété 5.3.3
Soient $\mathcal{R}$ une relation sans rangées de zéros ni rangées de uns, $\mathcal{L}(\mathcal{R})$ le treillis des concepts associé et $G_{\mathcal{R}}$ le graphe sous-jacent.
- *Si $A \times B$ est un atome de $\mathcal{L}(\mathcal{R})$ alors A est un moplex de $G_{\mathcal{R}}$;*
- *Si $A \times B$ est un co-atome de $\mathcal{L}(\mathcal{R})$ alors B est un moplex de $G_{\mathcal{R}}$;*
- *Il n'y a pas d'autre moplex dans $G_{\mathcal{R}}$.*

Preuve : Soit $C = (\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte, $\mathcal{L}(\mathcal{R})$ le treillis des concepts associé et $G_{\mathcal{R}} = (V, E)$ le graphe co-biparti sous-jacent.

1. Soit $A \times B$ un atome de $\mathcal{L}(\mathcal{R})$ représenté par le séparateur minimal $S = N(A) = N(B)$. Comme partie de $\mathscr{P}$, A est une clique. Nous pouvons affirmer que A est un module : supposons que ce ne soit pas le cas. Par la définition 1.3.5, il doit exister x et y dans A tels que $N(x) \neq N(y)$. On peut alors supposer sans perte de généralité qu'il existe un sommet b de $\mathscr{O}$ tel que $xb \in E$ et $yb \notin E$. Par conséquent, $A \times B$ a un prédécesseur dont l'intension est $A' \subseteq A - \{x\}$ et dont l'extension est $B' \supseteq B \cup \{b\}$. De plus, comme $y \in A'$ et que $\mathcal{R}$ n'a pas de rangées de uns, $A' \times B'$ ne peut pas être le bottom $\emptyset \times \mathscr{O}$ de $\mathcal{R}$, ce qui empêche $A \times B$ d'être un atome et provoque une contradiction.
   Par une preuve similaire, on montre dualement que si $A \times B$ est un co-atome de $\mathcal{L}(\mathcal{R})$ alors B est un moplex.

2. Réciproquement, soient A un moplex de $G_{\mathcal{R}}$ et $S = N(A)$ le séparateur minimal associé. Par la caractérisation 4.1.10, S définit deux composantes connexes, l'une d'elle étant A ; l'autre composante connexe sera notée B, avec $N(B) = S$.
   Si $A \subseteq \mathscr{P}$, par la caractérisation 4.1.13, $A \times B$ est un concept de $\mathcal{L}(\mathcal{R})$. Comme $A \neq \emptyset$ et que $\mathcal{R}$ n'a pas de rangées de zéros, $A \times B$ n'est pas le bottom de $\mathcal{L}(\mathcal{R})$ et a donc au moins un prédécesseur. Comme A est un module, pour tous sommets x et y de A on a $N(x) = N(y)$. Par conséquent, la seule manière d'étendre B de manière à obtenir un prédécesseur de $A \times B$ dans le treillis est de retirer tous les sommets de A, ce qui ne peut aboutir qu'à $\emptyset \times \mathscr{O}$, le bottom de $\mathcal{L}(\mathcal{R})$. Ainsi $A \times B$ est un atome de $\mathcal{L}(\mathcal{R})$.
   Si $A \subseteq \mathscr{O}$, on prouve dualement que $B \times A$ est un co-atome de $\mathcal{L}(\mathcal{R})$.

◇

Dans le cas d'un graphe co-biparti, les moplex correspondent exactement aux maxmods non dominants du graphe :

Propriété 5.3.4
Soit $G = (V, E)$ un graphe co-biparti et $X \subset V$ un maxmod de G. X est un moplex si et seulement si X est non-dominant.

Preuve : Soit $G = (V, E)$ un graphe co-biparti et $X \subset V$ un maxmod de G.

$\Longrightarrow$

Supposons que X soit un moplex. Alors $N(X)$ est un séparateur minimal qui induit (par la propriété 4.1.10) une deuxième composante connexe, dont le voisinage est $N(X)$. Par conséquent, X est non-dominant.

$\Longleftarrow$

Supposons que X soit non-dominant. Alors $Y = V - (N(X) \cup X)$ a le même voisinage que X et donc $N(X)$ est un séparateur minimal. Par conséquent, X est un moplex.

$\diamond$

Exemple 5.3.5

Continuons avec la relation $\mathcal{R}$ de l'exemple 5.2.12 (page 119).

Le treillis des concepts $\mathcal{L}(\mathcal{R})$ associé est donné dans la figure 5.5.

Les atomes de $\mathcal{L}(\mathcal{R})$ sont : $ah \times 236$, $b \times 123$ $c \times 125$ et $d \times 145$. Dans $G_{\mathcal{R}}$ (figure 5.3), $\{a, h\}$, $\{b\}$, $\{c\}$ et $\{d\}$ sont des moplex. Les co-atomes sont : $abfgh \times 3$, $abcgh \times 2$ et $bcde \times 1$. Dans $G_{\mathcal{R}}$, $\{1\}$, $\{2\}$ et $\{3\}$ sont des moplex. En conclusion $G_{\mathcal{R}}$ a exactement pour moplex : $\{a, h\}$, b, c, d, 1, 2 et 3.

On peut vérifier avec l'ordre de domination de la figure 5.4 que ces moplex correspondent exactement aux maxmods non dominants de $G_{\mathcal{R}}$. Les autres maxmods e, f, g, 4, 5 et 6 sont dominants.

$\diamond$

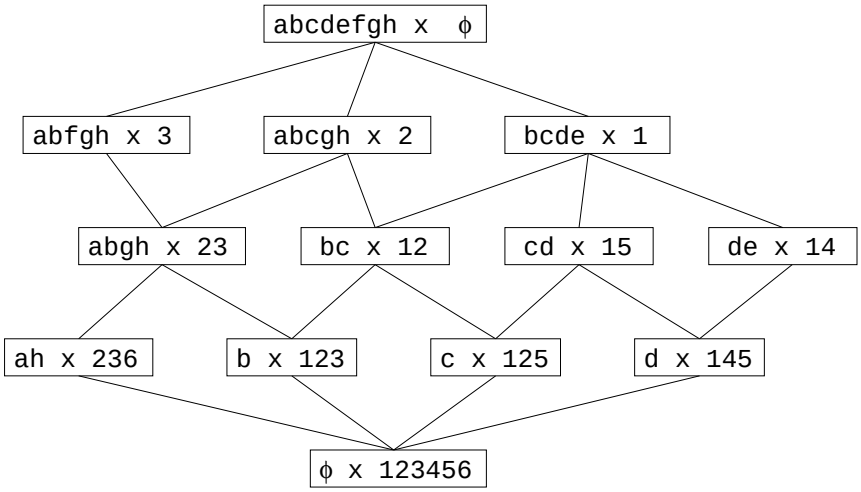


Fig. 5.5 – Le treillis $\mathcal{L}(\mathcal{R})$ de l'exemple 5.3.5.

5.3.3 Calcul de la couverture d'un élément

Chaque élément d'un treillis est le bottom du sous-treillis qu'il forme avec ses descendants. Si nous passons du treillis au sous-treillis, ce qui était la couverture de l'élément sélectionné devient la couche atomique du sous-treillis. Ainsi, les propriétés classiques sur les graphes vont nous permettre de caractériser les concepts qui forment la couverture d'un élément du treillis en nous appuyant sur la notion de domination dans un graphe.

Nous pouvons utiliser la propriété 5.3.4 et les résultats de la sous-section précédente pour calculer la couverture d'un élément $A \times B$ donné, en décomposant le treillis, obtenant ainsi un sous-treillis dont $A \times B$ est le bottom.

Théorème 5.3.6
Un concept $A' \times B'$ couvre un concept $A \times B$ si et seulement si il y a dans $G_1 = G(B \cup (\mathscr{P} - A))$ un maxmod non dominant X tel que $A' = X + A$.

Preuve : Soient $A \times B$ et $A' \times B'$ deux concepts d'un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$.

$\Longrightarrow$

Supposons que $A' \times B'$ couvre $A \times B$.
$A' \times B'$ est un atome du treillis dont $A \times B$ est le bottom, treillis des concepts de la sous-relation $\mathcal{R}_1 = \mathcal{R}(\mathscr{P} - A, B)$ dont le graphe associé est $G_1 = G(B \cup (\mathscr{P} - A))$. La propriété 5.3.3 nous permet d'affirmer que $X = A' - A$ est un moplex de G_1. La propriété 5.3.4 affirme alors que X est un maxmod non dominant de G_1.

$\Longleftarrow$

Supposons que $X = A' - A$ soit un maxmod non dominant de $G_1 = G(B \cup (\mathscr{P} - A))$. La propriété 5.3.4 montre que X est un moplex de G_1. $A \times B$ est le bottom du treillis des concepts de la sous-relation $\mathcal{R}_1 = \mathcal{R}(\mathscr{P} - A, B)$. On en déduit, par la propriété 5.3.3 que $A' \times B'$ est un atome de ce treillis. Par conséquent $A' \times B'$ couvre $A \times B$ dans $\mathcal{L}(\mathcal{R})$.

$\diamond$

Exemple 5.3.7
Dans le treillis $\mathcal{L}(\mathcal{R})$ de la figure 5.5, $abgh \times 23$ couvre $ah \times 236$. Dans le sous-graphe $G_2 = G(\{2, 3, 6\} \cup \{b, c, d, e, f, g\})$ de $G_{\mathcal{R}}$, $\{a, b, g, h\} - \{a, h\} = \{b, g\}$ est un maxmod non dominant. En effet, $\{a, h\}$ ayant disparu, b devient non dominant dans G_2 ; de plus la disparition de 1 permet à g de dominer en retour b, ce qui fait de $\{b, g\}$ un maxmod non dominant de G_2.

$\diamond$

Il faut remarquer, pour des questions de complexité, qu'un maxmod X de degré minimum est forcément non-dominant et que la recherche des sommets qui dominent un maxmod donné X peut être réalisée en temps linéaire par le calcul de l'ensemble des sommets qui sont universels dans $N(X)$.

Pour trouver l'ensemble des maxmods non-dominants de $G_{\mathcal{R}}$ nous proposons le procédé suivant :

Procédé algorithmique 5.3.8

Donnée : Le graphe $G_{\mathcal{R}}$ d'une relation binaire $\mathcal{R}$.
Résultat : L'ensemble des maxmods de $G_{\mathcal{R}}$.
Calculer les maxmods de $G_{\mathcal{R}}$ et les contracter ;
Tant que il reste un sommet dans le graphe **faire** :
  Choisir un sommet $\tilde{x}$ de degré minimum dans le graphe obtenu ainsi ;
  Calculer en temps linéaire l'ensemble des sommets qui dominent $\tilde{x}$;
  Retirer du graphe $\tilde{x}$ et les sommets qui le dominent.

Ce procédé algorithmique peut être implémenté en temps $O(m)$ par maxmod non-dominant calculé. Il sera utilisé dans la deuxième section du chapitre 6.

5.3.4 Comment un concept hérite de la domination de ses ancêtres

Un résultat qui sera essentiel dans les applications algorithmiques est la façon dont les relations de domination s'héritent le long d'une chaîne maximale du treillis. Si x domine y dans $G_{\mathcal{R}}(B \cup (\mathscr{P}-A))$ et si X est un maxmod non dominant de ce graphe, alors cette domination sera préservée dans le graphe correspondant au successeur $(A+X) \times (B-N^+(X))$ de $A \times B$, ainsi que dans tous ses descendants. D'autres dominations peuvent apparaître dans ce nouveau graphe $G_{\mathcal{R}}((B-N^+(X)) \cup (\mathscr{P}-(A+X)))$, mais x dominera encore y dans ce graphe. En remontant le long d'une chaîne du treillis, cette domination sera conservée jusqu'à ce que x et y appartiennent à un même maxmod non dominant qui servira alors à constituer un concept couvrant, ou jusqu'à ce que y soit utilisé pour définir un concept.

Propriété 5.3.9
Soit $A \times B$ un concept et $(A+X) \times (B-N^+(X))$ un concept qui le couvre. Soient $x, y \in \mathscr{P}-(A+X)$ tels que x domine y dans $G_1 = G_{\mathcal{R}}(B \cup (\mathscr{P}-A))$. Alors, x domine y dans $G_2 = G_{\mathcal{R}}((B-N^+_{G_1}(X)) \cup (\mathscr{P}-(A+X)))$.

Preuve : Soit $A \times B$ un concept et $(A+X) \times (B-N^+(X))$ un concept qui le couvre. Soient $x, y \in \mathscr{P}-A$ tels que x domine y dans $G_1 = G_{\mathcal{R}}(B \cup (\mathscr{P}-A))$.
x domine y dans G_1 donc $N^+_{G_1}(y) \subseteq N^+_{G_1}(x)$.
On a donc $(N^+_{G_1}(y)-N^+_{G_1}(X)) \subseteq (N^+_{G_1}(x)-N^+_{G_1}(X))$.
Autrement dit, avec $G_2 = G_{\mathcal{R}}((B-N^+_{G_1}(X)) \cup (\mathscr{P}-(A+X)))$, on a $N^+_{G_2}(y) \subseteq N^+_{G_2}(x)$ et x domine y dans G_2.
$\diamond$

5.4 La table de domination

Le calcul de la relation de domination sur un graphe se fait en $O(nm)$. Dans les applications algorithmiques, dont nous verrons deux exemples au chapitre 6, il arrive souvent qu'il faille mettre à jour la relation par des ajouts ou des retraits de sommets ou d'arêtes. Dans bien des cas, cette mise à jour peut se faire sans qu'il soit pour autant nécessaire de recalculer entièrement la relation de domination.

Nous proposons dans cette section une nouvelle structure de données, la table de domination, qui nous permet d'effectuer une mise à jour en $O(n)$ par opération élémentaire d'ajout ou de retrait.

Nous ne traiterons ici que la domination entre propriétés ; les mêmes techniques s'appliquent à la domination entre objets.

5.4.1 Représentation de la domination par une table

Pour maintenir les informations sur la domination entre propriétés, nous utilisons une table de domination L_{dom} qui, pour chaque paire (x, y) de propriétés, stocke les objets dont la présence **empêche** x de dominer y. Si E est l'ensemble des arêtes de $G_{\mathcal{R}}$, la liste $\mathrm{L}_{dom}[x, y]$ contiendra donc tous les objets i tels que $xi \notin E$ et $yi \in E$, c'est-à-dire $(x, i) \in \mathcal{R}$ et $(y, i) \notin \mathcal{R}$.

Exemple 5.4.1
Table de domination des propriétés L_{dom} correspondant à la relation $\mathcal{R}$ de l'exemple 5.2.12 (page 119) :

L_{dom}	a	b	c	d	e	f	g	h
a	$\emptyset$	$\{1\}$	$\{1,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\emptyset$	$\emptyset$
b	$\{6\}$	$\emptyset$	$\{5\}$	$\{4,5\}$	$\{4\}$	$\emptyset$	$\emptyset$	$\{6\}$
c	$\{3,6\}$	$\{3\}$	$\emptyset$	$\{4\}$	$\{4\}$	$\{3\}$	$\{3\}$	$\{3,6\}$
d	$\{2,3,6\}$	$\{2,3\}$	$\{2\}$	$\emptyset$	$\emptyset$	$\{3\}$	$\{2,3\}$	$\{2,3,6\}$
e	$\{2,3,6\}$	$\{2,3\}$	$\{2,5\}$	$\{5\}$	$\emptyset$	$\{3\}$	$\{2,3\}$	$\{2,3,6\}$
f	$\{2,6\}$	$\{1,2\}$	$\{1,2,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\{2\}$	$\{2,6\}$
g	$\{6\}$	$\{1\}$	$\{1,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\emptyset$	$\{6\}$
h	$\emptyset$	$\{1\}$	$\{1,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\emptyset$	$\emptyset$

L'ordre de domination correspondant est représenté dans la figure 5.4 (page 120).
$\diamond$

Le procédé algorithmique de construction de la table de domination L_{dom} est le suivant :

Algorithme CONSTRUCTION DE LA TABLE DE DOMINATION ENTRE PROPRIÉTÉS :

Donnée : Un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ et son graphe sous-jacent $G_{\mathcal{R}}$.
Résultat : La table de domination entre propriétés L_{dom}.
Initialisation : La table L_{dom} contient des listes vides.
Début
Pour x appartenant à $\mathscr{P}$ **faire :**
  Pour y appartenant à $\mathscr{P}$ **faire :**
    Pour i appartenant à $N^+(y)$ **faire :**
    // *c'est-à-dire Pour* $i \in \mathscr{O}$ *tel que* yi *soit une arête de* E
      Si $xi \notin G_{\mathcal{R}}$ **alors**
        Ajouter i à $\mathrm{L}_{dom}[x, y]$;
Fin.

La table de domination L_{dom} a une taille globale en $O(nm)$. En effet, pour chaque propriété (colonne) x, on examine le voisinage parmi les objets de chaque propriété (ligne) y ; chaque zéro de $\mathcal{R}$ sera donc examiné une fois pour chaque colonne et il y a n colonnes. La complexité de l'algorithme ci-dessus est donc en $O(nm)$.

Notons qu'on a $\mathrm{L}_{dom}[x, x] = \emptyset$ pour tout x.

5.4.2 Requêtes sur la table de domination

Pour vérifier si une propriété x domine une propriété y, il suffit de vérifier dans la table de domination si la liste $\mathrm{L}_{dom}[x, y]$ est vide (aucun objet n'empêche alors x de dominer y). Par conséquent, les listes vides d'une colonne x donneront exactement les propriétés dominées par x.

La table de domination L_{dom} permet de retrouver l'ordre de domination sur les propriétés : deux propriétés x et y appartiennent à un même maxmod si et seulement si $\mathrm{L}_{dom}[x, y]$=$\mathrm{L}_{dom}[y, x] = \emptyset$. Un maxmod représenté par une propriété x dominera un maxmod représenté par une propriété y si et seulement si $\mathrm{L}_{dom}[x, y] = \emptyset$.

Exemple 5.4.2
De la table L_{dom} de l'exemple 5.4.1, on peut déduire :

- g domine b puisque $\mathrm{L}_{dom}[g, b] = \emptyset$. Par contre b ne domine pas g puisque $\mathrm{L}_{dom}[b, g] = \{1\}$. L'objet 1 empêche b de dominer g ; la disparition de 1 dans le sous-graphe $G_2 = G(\{2, 3, 6\} \cup \{b, c, d, e, f, g\})$ de $G_{\mathcal{R}}$, on l'a vu dans l'exemple 5.3.7, permettra à b de dominer g, formant un maxmod non dominant $\{b, g\}$ de G_2.
- $\mathrm{L}_{dom}[a, h] = \emptyset$=$\mathrm{L}_{dom}[h, a]$, donc a domine h et réciproquement, $\{a, h\}$ forme un maxmod de $G_{\mathcal{R}}$. Dans L_{dom}, les deux colonnes a et h sont identiques, en termes de voisinage, on a $N^+(a) = N^+(h)$. Les seules cases des colonnes a et h dont les listes sont vides sont celles des lignes a et h, ainsi le maxmod $\{a, h\}$ est non dominant.
- La colonne g peut être obtenue par intersection des colonnes a et b. Cela signifie que la relation $\mathcal{R}$ n'est pas réduite : $N^+(g) = N^+(a) \cap N^+(b)$; g n'apparaîtra dans l'intension d'un concept que si a et b y apparaissent aussi.

◇

5.4.3 Mise à jour de la table de domination

L'ajout ou le retrait d'une croix dans la relation est susceptible de modifier les rapports de domination entre sommets. Les algorithmes suivants permettent, dans un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$, de mettre à jour la table L_{dom}, en $O(|\mathscr{P}|)$ par arête modifiée dans le graphe $G_{\mathcal{R}}$ sous-jacent. Pour retirer ou ajouter un sommet, on pourra retirer ou ajouter toutes les croix impliquant ce sommet.

Algorithme `AjouterCroix` :

Données : La table L_{dom} et l'arête xi à supprimer de $G_{\mathcal{R}}$.
Résultat : La table mise à jour pour le nouveau contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R}+\{(x, i)\})$.
Début
Pour chaque propriété $y \neq x$ **faire :**
  Si $yi \in G_{\mathcal{R}}$
    alors Ajouter i à $\mathrm{L}_{dom}[x, y]$; // *Ajouts dans la colonne de x.*
    sinon Supprimer i de $\mathrm{L}_{dom}[y, x]$; // *Suppressions dans la ligne de x.*
Fin.

Algorithme `RetirerCroix` :

Données : La table L_{dom} et l'arête xi à ajouter à $G_{\mathcal{R}}$.
Résultat : La table mise à jour pour le nouveau contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R}-\{(x,i)\})$.
Début
Pour chaque propriété $y \neq x$ **faire :**
  Si $yi \in G_{\mathcal{R}}$
    alors Supprimer i de $\mathrm{L}_{dom}[x,y]$; // *Suppressions dans la colonne de x.*
    sinon Ajouter i à $\mathrm{L}_{dom}[y,x]$; // *Ajouts dans la ligne de x.*
Fin.

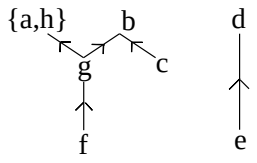


FIG. 5.6 – L'ordre de domination après ajout de $(b,5)$ à $\mathcal{L}(\mathcal{R})$ dans l'exemple 5.4.3.

Exemple 5.4.3

• Ajoutons $(b,5)$ à la relation $\mathcal{R}$ de l'exemple 5.2.12 dont la table de domination figure dans l'exemple 5.4.1.
On obtient une nouvelle relation $\mathcal{R}'$ définie par la table suivante :

$\mathcal{R}'$	a	b	c	d	e	f	g	h
1		×	×	×	×			
2	×	×	×				×	×
3	×	×				×	×	×
4				×	×			
5		×	×	×				
6	×							×

Dans $\mathcal{R}$, il y avait à la ligne 5 les croix c et d et les "non-croix" a, e, f, g et h. Il faut donc, dans L_{dom}, ajouter 5 aux listes $[b,a]$, $[b,e]$, $[b,f]$, $[b,g]$ et $[b,h]$ (colonne de b) et le retirer des listes $[c,b]$ et $[d,b]$ (ligne de b).

On obtient, pour $\mathcal{R}'$, une nouvelle table de domination L'_{dom} :

L'_{dom}	a	b	c	d	e	f	g	h
a	$\emptyset$	$\{1,\mathbf{5}\}$	$\{1,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\emptyset$	$\emptyset$
b	$\{6\}$	$\emptyset$	$\{\not{5}\}$	$\{4,\not{5}\}$	$\{4\}$	$\emptyset$	$\emptyset$	$\{6\}$
c	$\{3,6\}$	$\{3\}$	$\emptyset$	$\{4\}$	$\{4\}$	$\{3\}$	$\{3\}$	$\{3,6\}$
d	$\{2,3,6\}$	$\{2,3\}$	$\{2\}$	$\emptyset$	$\emptyset$	$\{3\}$	$\{2,3\}$	$\{2,3,6\}$
e	$\{2,3,6\}$	$\{2,3,\mathbf{5}\}$	$\{2,5\}$	$\{5\}$	$\emptyset$	$\{3\}$	$\{2,3\}$	$\{2,3,6\}$
f	$\{2,6\}$	$\{1,2,\mathbf{5}\}$	$\{1,2,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\{2\}$	$\{2,6\}$
g	$\{6\}$	$\{1,\mathbf{5}\}$	$\{1,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\emptyset$	$\{6\}$
h	$\emptyset$	$\{1,\mathbf{5}\}$	$\{1,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\emptyset$	$\emptyset$

Dans la nouvelle relation $\mathcal{R}'$, c domine maintenant b. La figure 5.6 donne le nouvel ordre de domination.

• Supprimons maintenant $(b, 1)$ de cette dernière relation $\mathcal{L}(\mathcal{R}')$.
La nouvelle relation $\mathcal{L}(\mathcal{R}'')$ est définie par la table suivante :

$\mathcal{L}(\mathcal{R}'')$	a	b	c	d	e	f	g	h
1			×	×	×			
2	×	×	×				×	×
3	×	×				×	×	×
4				×	×			
5		×	×	×				
6	×							×

Dans $\mathcal{R}''$ il y avait à la ligne 1 les croix c, d et e et les "non-croix" a, f, g et h. 1 doit donc être retiré des listes $[b, a]$, $[b, f]$, $[b, g]$ et $[b, h]$ et ajouté aux listes $[c, b]$, $[d, b]$ et $[e, b]$.

On obtient, pour $\mathcal{R}''$, une nouvelle table de domination :

	a	b	c	d	e	f	g	h
a	$\emptyset$	$\{\not{1}, 5\}$	$\{1, 5\}$	$\{1, 4, 5\}$	$\{1, 4\}$	$\emptyset$	$\emptyset$	$\emptyset$
b	$\{6\}$	$\emptyset$	$\{\mathbf{1}\}$	$\{\mathbf{1}, 4\}$	$\{\mathbf{1}, 4\}$	$\emptyset$	$\emptyset$	$\{6\}$
c	$\{3, 6\}$	$\{3\}$	$\emptyset$	$\{4\}$	$\{4\}$	$\{3\}$	$\{3\}$	$\{3, 6\}$
d	$\{2, 3, 6\}$	$\{2, 3\}$	$\{2\}$	$\emptyset$	$\emptyset$	$\{3\}$	$\{2, 3\}$	$\{2, 3, 6\}$
e	$\{2, 3, 6\}$	$\{2, 3, 5\}$	$\{2, 5\}$	$\{5\}$	$\emptyset$	$\{3\}$	$\{2, 3\}$	$\{2, 3, 6\}$
f	$\{2, 6\}$	$\{\not{1}, 2, 5\}$	$\{1, 2, 5\}$	$\{1, 4, 5\}$	$\{1, 4\}$	$\emptyset$	$\{2\}$	$\{2, 6\}$
g	$\{6\}$	$\{\not{1}, 5\}$	$\{1, 5\}$	$\{1, 4, 5\}$	$\{1, 4\}$	$\emptyset$	$\emptyset$	$\{6\}$
h	$\emptyset$	$\{\not{1}, 5\}$	$\{1, 5\}$	$\{1, 4, 5\}$	$\{1, 4\}$	$\emptyset$	$\emptyset$	$\emptyset$

Après cette deuxième modification de $\mathcal{R}$, c ne domine plus b. L'ordre de domination retrouve sa forme initiale (figure 5.4, page 120).
◇

Cette technique de mise à jour pourrait être étendue à la construction efficace de la table de domination initiale dans le cas d'une relation très dense, en considérant la relation avec seulement une diagonale de zéros (ce qui correspond à un graphe sans domination) puis en retirant des éléments jusqu'à ce que la relation désirée soit obtenue.

5.4.4 Une table simplifiée : la table de cardinalité

Dans bien des cas, pour répondre aux requêtes, il est suffisant de connaître la **taille** de chaque liste $\mathrm{L}_{dom}[y, x]$, ce qui présente l'avantage de réduire l'espace nécessaire à $O(n^2)$ au lieu de $O(nm)$.

Un maxmod X est non dominant quand, pour un quelconque de ses éléments x, le nombre de sommets que x domine est exactement $|X|$. Nous allons donc utiliser une *table de cardinalité* $T_{\mathscr{P}}$ qui contiendra des nombres compris entre 0 et $|\mathscr{O}|$, la case $T_{\mathscr{P}}[x, y]$ contenant la taille de la liste $L_{dom}[x, y]$, c'est-à-dire le nombre de sommets qui empêchent x de dominer y. En particulier, le sommet x dominera le sommet y si et seulement si $T_{\mathscr{P}}[x, y] = 0$; afin d'avoir un accès rapide à cette information, nous conservons aussi un tableau $D_{\mathscr{P}}$ indicé par les éléments de $\mathscr{P}$ où $D_{\mathscr{P}}[x]$ donne le nombre de sommets y tels que $T_{\mathscr{P}}[x, y] = 0$, c'est-à-dire le nombre de sommets que x domine. Ainsi un maxmod

X sera non-dominant si et seulement si, pour $x \in X$ quelconque, $D_{\mathscr{P}}[x] = |\tilde{x}|$, où $\tilde{x}$ est le maxmod contenant x ; la question "*Quels sont les maxmods non dominants*" pourra obtenir une réponse rapide (en un temps $O(n)$) avec le tableau $D_{\mathscr{P}}$. Nous pourrons de même répondre à la question "*Quels sont les maxmods non dominés*".

Le procédé pour construire directement la table de domination initiale $T_{\mathscr{P}}$ (à partir d'une table initialisée avec des zéros dans chaque case) puis le tableau $D_{\mathscr{P}}$ (initialisé pareillement) est le suivant :

Algorithme 5.4.4 CONSTRUCTION DE LA TABLE DE CARDINALITÉ :

Début
// *Construire* $T_{\mathscr{P}}$
Pour chaque x dans $\mathscr{P}$ **faire :**
  Pour chaque y dans $\mathscr{P}$ **faire :**
    Pour chaque i dans $\mathscr{O}$ **faire :**
      Si $(x, i) \in \mathcal{R}$ et $(y, i) \notin \mathcal{R}$ **alors** ajouter 1 à $T_{\mathscr{P}}[x, y]$;
// *Construire* $D_{\mathscr{P}}$
  Pour chaque y dans $\mathscr{P}$ **faire :**
    Si $T_{\mathscr{P}}[x, y] = 0$ alors ajouter 1 à $D_{\mathscr{P}}[x]$;
Fin.

Exemple 5.4.5
La table de cardinalité $T_{\mathscr{P}}$ et le tableau $D_{\mathscr{P}}$ associés à la table de domination présentée dans l'exemple 5.4.1 sont les suivants :

$T_{\mathscr{P}}$	a	b	c	d	e	f	g	h
a	0	1	2	3	2	0	0	0
b	1	0	1	2	1	0	0	1
c	2	1	0	1	1	1	1	2
d	3	2	1	0	0	1	2	3
e	3	2	2	1	0	1	2	3
f	2	2	3	3	2	0	1	2
g	1	1	2	3	2	0	0	1
h	0	1	2	3	2	0	0	0

$D_{\mathscr{P}}$

a	b	c	d	e	f	g	h
2	1	1	1	2	5	4	2

◇

Les tables de domination ont été utilisées pour la conception de deux applications algorithmiques qui sont présentées dans le chapitre suivant.

Chapitre 6

Applications algorithmiques

Dans ce chapitre, nous étudions deux problèmes algorithmiques : le maintien d'une sous-hiérarchie de Galois et la génération d'un treillis des concepts. Ces deux applications illustrent l'intérêt algorithmique des outils théoriques que nous avons présentés dans les chapitres 4 et 5.

- La première application a été présentée, en collaboration avec Anne Berry dans le workshop MASPEGHI, organisé par Marianne Huchard dans le cadre de la conférence OOIS'02 (Object-Oriented Information System) et publié dans LNCS (voir [BS02c]).

- La deuxième, faite en collaboration avec Anne Berry et Jean-Paul Bordat, est en cours de soumission (voir [BBS02]).

6.1 Maintien d'une sous-hiérarchie de Galois

Dans le cadre des applications aux langages à objets, en génie logiciel, on s'intéresse à la construction, à la réorganisation et au maintien d'une hiérarchie d'héritage. L'organisation en treillis de Galois intéresse les chercheurs dans ce domaine car elle permet une factorisation maximale des propriétés. Etant donné la taille trop importante de ce treillis, qui le rend difficile à visualiser par l'utilisateur, une des solutions est de travailler sur ce qu'on appelle la sous-hiérarchie de Galois. Cette sous-hiérarchie contient sous une forme simplifiée les concepts particuliers qui introduisent une propriété ou un objet (pour plus de précision, voir [HDL00], [GM93], [CL96] et [Huch02]). Cette structure nous intéresse particulièrement car, comme nous allons le voir, elle offre des similitudes frappantes avec les ordres de domination présentés au chapitre 5.

6.1.1 Treillis des concepts et sous-hiérarchies de Galois

Pour améliorer la lisibilité des treillis, certains auteurs sont amenés à simplifier les étiquettes des éléments : seuls sont conservés les objets ou les propriétés qui y apparaissent pour la première fois, de manière ascendante pour les propriétés et de manière descendante pour les objets.

Définition 6.1.1 INTRODUCTEUR.
*Nous dirons qu'un concept $A \times B$ est l'**introducteur** d'une propriété $x \in A$ quand, dans $\mathcal{L}(\mathcal{R})$, x est absent de tous les ascendants de $A \times B$.*
*Nous dirons qu'un concept $A \times B$ est l'**introducteur** d'un objet $i \in B$ quand, dans $\mathcal{L}(\mathcal{R})$, i est absente de tous les descendants de $A \times B$.*
Nous dirons qu'un concept est introducteur s'il existe une propriété ou un objet dont il est l'introducteur.

Définition 6.1.2 TREILLIS SIMPLIFIÉ.
*Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte et $\mathcal{L}(\mathcal{R})$ le treillis des concepts associé. On obtient un **treillis simplifié** de $\mathcal{L}(\mathcal{R})$ en remplaçant l'étiquette $A \times B$ de chaque concept par l'étiquette (G, D), où G contient les propriétés dont ce concept est l'introducteur et D contient les objets de B dont ce concept est l'introducteur.*

Notons que si un concept n'est pas introducteur, il est simplifié en $(\emptyset, \emptyset)$. Une deuxième simplification élimine ces étiquettes et fournit un ordre appelé sous-hiérarchie de Galois :

Définition 6.1.3 SOUS-HIÉRARCHIE DE GALOIS.
*Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte et $\mathcal{L}(\mathcal{R})$ le treillis des concepts associé. On appelle **sous-hiérarchie de Galois** l'ordre, qu'on notera $\mathcal{H}(\mathcal{R})$, obtenu à partir du treillis simplifié issu de $\mathcal{L}(\mathcal{R})$ en supprimant le top, le bottom ainsi que tous les éléments étiquetés par $(\emptyset, \emptyset)$.*

Remarque 6.1.4
Il est clair que ce procédé supprime tous les éléments de $\mathcal{L}(\mathcal{R})$ qui ne sont pas introducteurs. Dans cette optique, puisqu'on ne considère que des relations sans rangées de uns ou de zéros (leur cas a été réglé au chapitre 4), il est cohérent de supprimer aussi le top et le bottom.
Remarquons aussi que, si la relation est réduite, $\mathcal{H}(\mathcal{R})$ contient exactement les éléments irréductibles de $\mathcal{L}(\mathcal{R})$ puisque, dans ce cas, les irréductibles sont exactement les éléments introducteurs. En effet, pour calculer la relation réduite dont un treillis est treillis de Galois, l'algorithme 1.2.46 (page 31) met en bijection l'ensemble des irréductibles du treillis avec l'ensemble des propriétés et objets de la relation construite.

Par construction, le nombre d'éléments dans une sous-hiérarchie de Galois ne dépasse pas $n = |\mathscr{P}| + |\mathscr{O}|$.

Exemple 6.1.5
Reprenons la relation $\mathcal{R}$ de l'exemple 5.2.12 du chapitre 5 (page 119).

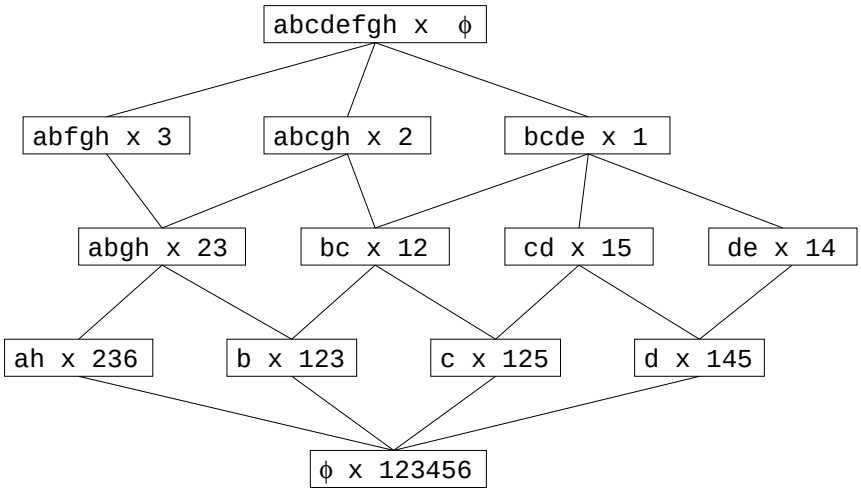


Fig. 6.1 – Treillis $\mathcal{L}(\mathcal{R})$ associé à la relation $\mathcal{R}$ de l'exemple 6.1.5.

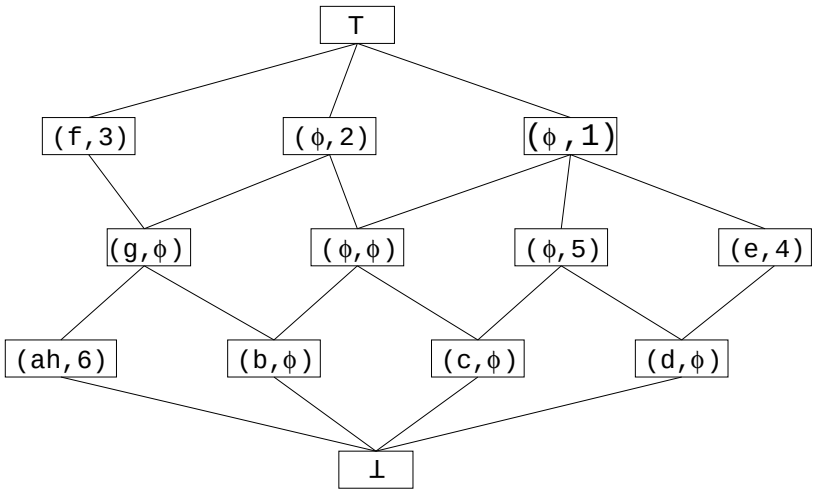


Fig. 6.2 – Treillis simplifié associé à $\mathcal{L}(\mathcal{R})$ de la figure 6.1.

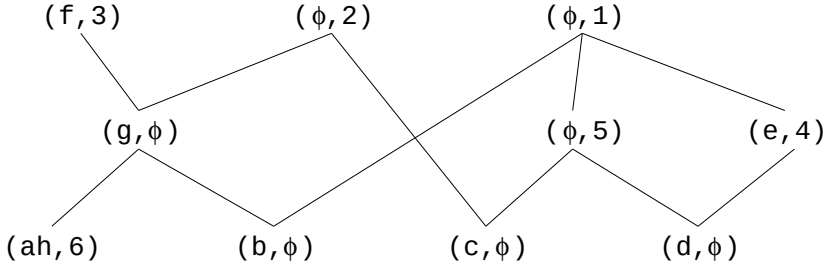


Fig. 6.3 – Sous-hiérarchie de Galois associé à la relation $\mathcal{R}$ de l'exemple 6.1.5.

$\mathcal{R}$	a	b	c	d	e	f	g	h
1		×	×	×	×			
2	×	×	×				×	×
3	×	×				×	×	×
4				×	×			
5			×	×				
6	×							×

FIG. 6.4 – Table de la relation $\mathcal{R}$ de l'exemple 6.1.5.

Nous redonnons la table de $\mathcal{R}$ dans la figure 6.4 et le treillis $\mathcal{L}(\mathcal{R})$ associé dans la figure 6.1.

Le treillis simplifié correspondant est présenté dans la figure 6.2 et la sous-hiérarchie de Galois $\mathcal{H}(\mathcal{R})$ dans la figure 6.3. Notons que l'ordre partiel $\mathcal{H}(\mathcal{R})$ n'est pas un treillis puisque les éléments $(\emptyset, 2)$ et $(\emptyset, 1)$ n'ont pas de borne inférieure.
Le concept $abgh \times 23$ est l'introducteur de la propriété g, il est simplifié en $(g, \emptyset)$ dans $\mathcal{H}(\mathcal{R})$. Le concept $bcde \times 1$ est l'introducteur de l'objet 1, il est simplifié en $(\emptyset, 1)$ dans $\mathcal{H}(\mathcal{R})$. Le concept $ah \times 236$ est l'introducteur des propriétés a et h et des objets 3 et 6, il est simplifié en $(ah, 36)$ dans $\mathcal{H}(\mathcal{R})$. Le concept $bc \times 12$ n'est pas introducteur, il est absent de $\mathcal{H}(\mathcal{R})$.
La relation $\mathcal{R}$ n'est pas réduite car la colonne de g est l'intersection de celles de a et de b ; le concept introducteur de g, $abgh \times 23$, n'est pas un irréductible de $\mathcal{L}(\mathcal{R})$.
◇

6.1.2 Utilisation de la domination pour décomposer une sous-hiérarchie de Galois

Au chapitre 5, nous avons séparé les propriétés des objets pour travailler sur l'ordre de domination. Nous introduisons donc une décomposition de la sous-hiérarchie de Galois en deux sous-hiérarchies, l'une construite sur les intensions, l'autre sur les extensions. Chacune de ces sous-hiérarchies sera construite en conservant respectivement la partie gauche et la partie droite des étiquettes de la sous-hiérarchie de Galois, puis, en cohérence avec la définition des sous-hiérarchies, en retirant dans chacune des nouvelles structures obtenues les éléments dont l'étiquette est vide.

La sous-hiérarchie sur les intensions sera définie comme suit :

Définition 6.1.6 SOUS-HIÉRARCHIE SUR LES INTENSIONS.
Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte et $\mathcal{H}(\mathcal{R})$ la sous-hiérarchie de Galois associée. Nous appellerons ***sous-hiérarchie sur les intensions*** *l'ordre obtenu à partir de $\mathcal{H}(\mathcal{R})$ en ne conservant, pour chaque élément (G, D), que la partie gauche G de son étiquette, puis en supprimant tous les éléments étiquetés par $\emptyset$. Ce nouvel ordre sera noté $\mathcal{I}(\mathcal{R})$.*

Nous définirons dualement une sous-hiérarchie sur les extensions, que nous noterons $\mathcal{E}(\mathcal{R})$. Nous ne traiterons dans ce qui suit que le cas de la sous-hiérarchie sur les intensions ;

la sous-hiérarchie sur les extensions s'étudie de façon similaire.

Lemme 6.1.7
Un concept $A \times B$ est l'introducteur d'une propriété $x \in A$ si et seulement si $N^+(x) = \mathscr{O} - B$.

Preuve : Soient $A \times B$ un concept d'un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ et $x \in A$.

$\Longrightarrow$

On suppose que $A \times B$ est l'introducteur de x. Par définition d'un concept, $B \subseteq N^+(x)$. Supposons qu'il existe $i \in (\mathscr{O} - B) - N^+(x)$; on peut donc construire un autre concept $A' \times B'$ dont l'intension A' contient x et dont l'extension B' inclut $B + \{i\}$. Puisque $B \subset B + \{i\} \subseteq B'$, $A' \times B'$ est un ascendant de $A \times B$ dont l'intension contient x. Par conséquent $A \times B$ ne peut être l'introducteur de x, ce qui est contradictoire.

$\Longleftarrow$

On suppose que $N^+(x) = \mathscr{O} - B$. Soit $A' \times B'$ un ascendant de $A \times B$. Par définition d'un concept, on a $B \subset B'$. Puisque $N^+(x) = \mathscr{O} - B$, alors $\forall i \in B' - B$, $(x, i) \notin \mathcal{R}$. Par conséquent, x ne avoir $A' \times B'$ pour introducteur, ni aucun autre ascendant de $A \times B$, il apparaît pour la première fois dans $A \times B$ qui en est donc l'introducteur.

$\diamond$

Lemme 6.1.8
Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte et $G_{\mathcal{R}}$ son graphe sous-jacent.
Les éléments de la sous-hiérarchie sur les intensions $\mathcal{I}(\mathcal{R})$ sont exactement étiquetés par les maxmods de $G_{\mathcal{R}}(\mathscr{P})$.

Preuve : Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte, $\mathcal{L}(\mathcal{R})$ son treillis des concepts et $G_{\mathcal{R}}$ son graphe sous-jacent.
Chaque propriété x de $\mathscr{P}$ se retrouvera dans au moins un des éléments de la sous-hiérarchie sur les intensions $\mathcal{I}(\mathcal{R})$: par construction d'un treillis des concepts, chaque propriété apparaît dans au moins un élément de $\mathcal{L}(\mathcal{R})$, il y aura donc un concept qui sera l'introducteur de x et ce concept se retrouvera, avec une étiquette gauche non vide, dans la sous-hiérarchie de Galois $\mathcal{H}(\mathcal{R})$ et donc dans $\mathcal{I}(\mathcal{R})$. Par ailleurs la propriété 5.2.3 et le lemme 6.1.7 nous assurent que tous les sommets d'un même maxmod de $G_{\mathcal{R}}$ se retrouveront dans le même élément de $\mathcal{I}(\mathcal{R})$: si x et y appartiennent à un même maxmod de $G_{\mathcal{R}}$ alors $N^+(x) = N^+(y)$, et de ce fait x et y auront le même introducteur dans $\mathcal{H}(\mathcal{R})$ et donc dans $\mathcal{I}(\mathcal{R})$.
D'autre part, si deux propriétés x et y n'appartiennent pas à un même maxmod de $G_{\mathcal{R}}$ alors $N^+(x) \neq N^+(y)$ et par le lemme 6.1.7 ne peuvent avoir le même concept introducteur, ils ne peuvent donc pas se retrouver dans le même élément de $\mathcal{I}(\mathcal{R})$.

$\diamond$

Lemme 6.1.9
Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte et X_1 et X_2 deux éléments de $\mathcal{I}(\mathcal{R})$. $X_1 \leq X_2$ dans $\mathcal{I}(\mathcal{R})$ si et seulement si X_2 domine X_1.

Preuve : Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte, X_1 et X_2 deux éléments de sa sous-hiérarchie sur les intensions $\mathcal{I}(\mathcal{R})$.

Par le lemme 6.1.8, X_1 et X_2 sont des maxmods de $G_{\mathcal{R}}$. On peut donc choisir un représentant x_1 de X_1 et un représentant x_2 de X_2.
X_1 est associé à un concept $A_1 \times B_1$ de $\mathcal{L}(\mathcal{R})$, $A_1 \supseteq X_1$, introducteur de x_1, donc (lemme 6.1.7) $N^+(x_1) = \mathscr{O} - B_1$. Il existe de même un concept $A_2 \times B_2$, avec $A_2 \supseteq X_2$, introducteur de x_2, et on a $N^+(x_2) = \mathscr{O} - B_2$. Par le théorème 5.2.11, x_2 domine x_1 si et seulement si $N^+(x_1) \subseteq N^+(x_2)$, ce qui est équivalent à $\mathscr{O} - N^+(x_1) \supseteq \mathscr{O} - N^+(x_2)$ et donc à $B_1 \supseteq B_2$. Cette dernière inclusion est vraie, par construction des concepts, si et seulement si $A_1 \times B_1 \leq A_2 \times B_2$ dans $\mathcal{L}(\mathcal{R})$, ce qui est équivalent à $X_1 \leq X_2$ dans $\mathcal{I}(\mathcal{R})$.
◇

Remarque 6.1.10 *Le lemme 6.1.9 établit l'équivalence entre comparabilité de concepts dans $\mathcal{L}(\mathcal{R})$ et domination dans $G_{\mathcal{R}}$. Il peut être mis en parallèle avec le lemme 4.2.1 qui établit l'équivalence entre comparabilité de concepts dans $\mathcal{L}(\mathcal{R})$ et "non-croisement" des séparateurs correspondant dans $G_{\mathcal{R}}$. On pourrait ainsi relier les notions de domination et de séparation dans un graphe, ce qui aboutirait peut-être à des outils nouveaux qui permettraient d'améliorer l'énumération des séparateurs minimaux ou des ab-séparateurs minimaux (séparateurs qui sont minimaux pour la séparation d'une même paire de sommets donnée $\{a, b\}$). Cependant, cette étude dépasse le cadre de travail que nous nous sommes fixés.*

Théorème 6.1.11 SOUS-HIÉRARCHIE ET DOMINATION.
Soit $(\mathscr{P}, \mathscr{O}, \mathcal{R})$ un contexte et $G_{\mathcal{R}}$ son graphe sous-jacent.
La donnée de l'ordre de domination sur les propriétés de $G_{\mathcal{R}}$ est équivalent à la donnée de la sous-hiérarchie sur les intensions $\mathcal{I}(\mathcal{R})$ associée à $\mathcal{L}(\mathcal{R})$.

Preuve : Ce théorème découle directement des lemmes 6.1.8 et 6.1.9, en utilisant l'ordre de domination sur les maxmods de $G_{\mathcal{R}}$ défini dans le chapitre 5.
◇

Il suffit donc de conserver les informations de domination sur $G_{\mathcal{R}}$ pour avoir toutes les informations sur $\mathcal{I}(\mathcal{R})$. De plus, la taille de $\mathcal{I}(\mathcal{R})$ est bornée par $|\mathscr{P}|$.

Exemple 6.1.12
La sous-hiérarchie sur les intensions de l'exemple 6.1.5 est présentée dans la figure 6.6, la sous-hiérarchie sur les extensions dans la figure 6.7. L'ordre de domination correspondant est celui de la figure 5.4, il est rappelé dans la figure 6.5 ci-dessous.
◇

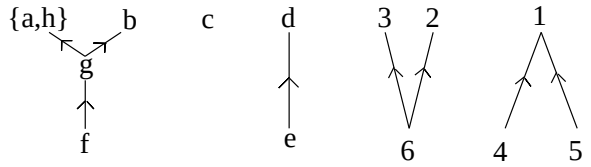
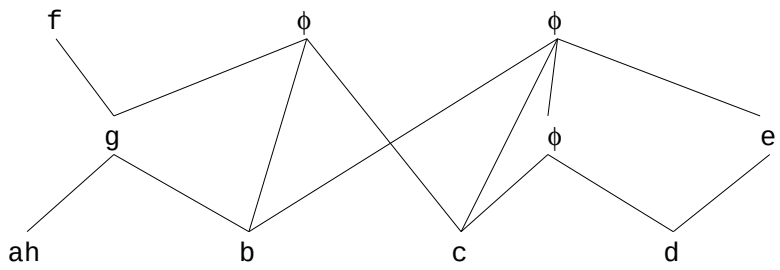


FIG. 6.5 – L'ordre de domination de l'exemple 6.1.12.

Première étape, simplification des étiquettes de $\mathcal{H}(\mathcal{R})$:



Deuxième étape, suppression des éléments étiquetés par $\emptyset$ pour obtenir $\mathcal{I}(\mathcal{R})$:

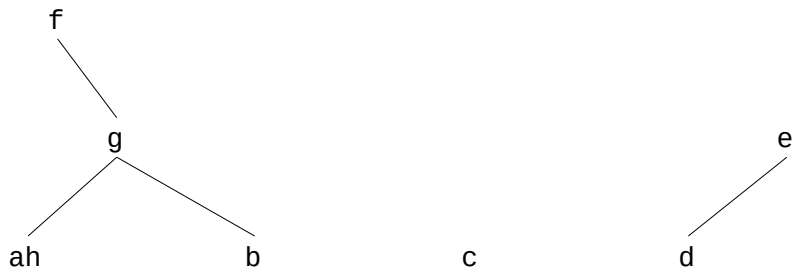
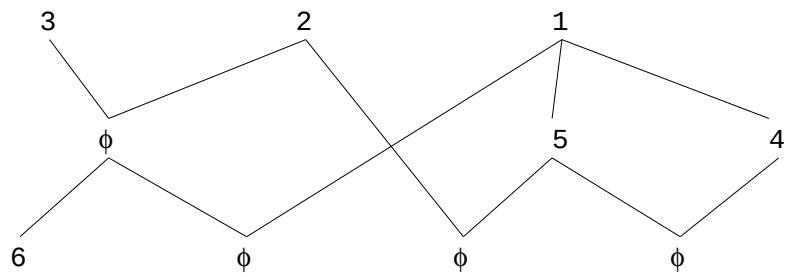


Fig. 6.6 – Construction de la sous-hiérarchie sur les intensions $\mathcal{I}(\mathcal{R})$ de l'exemple 6.1.12.

Première étape, simplification des étiquettes de $\mathcal{H}(\mathcal{R})$:



Deuxième étape, suppression des éléments étiquetés par $\emptyset$ pour obtenir $\mathcal{E}(\mathcal{R})$:

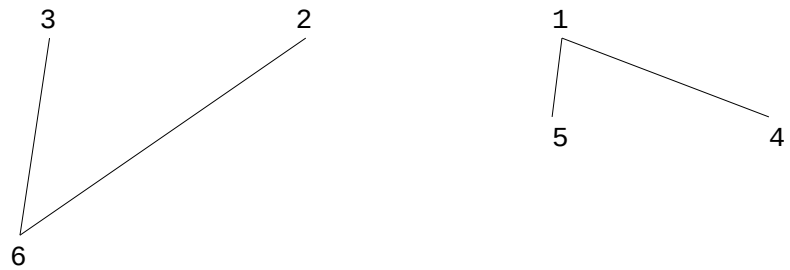


Fig. 6.7 – Construction de la sous-hiérarchie sur les extensions $\mathcal{E}(\mathcal{R})$ de l'exemple 6.1.12.

6.1.3 Mise à jour des informations de domination

Nous allons montrer ici comment on peut, à partir de la table de domination, reconstituer l'ordre de domination sur les propriétés et donc la sous-hiérarchie sur les intensions. Nous verrons ensuite comment retrouver la sous-hiérarchie de Galois à partir de la sous-hiérarchie sur les intensions et de celle sur les extensions.

Ces reconstitutions sont utiles quand la relation change. Nous avons vu au chapitre précédent que la table de domination peut être mise à jour pour prendre en compte les ajouts ou suppressions intervenues (algorithmes `AjouterCroix` et `RetirerCroix`). La nouvelle sous-hiérarchie de Galois pourra être reconstituée à partir de la table de domination modifiée.

Exemple 6.1.13
Nous allons successivement ajouter puis retirer une croix de la relation $\mathcal{R}$ de l'exemple 6.1.5.

• L'ajout de $(b, 5)$ à la relation $\mathcal{R}$, comme le faisait l'exemple 5.4.3, aboutit à une nouvelle relation $\mathcal{R}'$. Le treillis des concepts $\mathcal{L}(\mathcal{R}')$ associé à $\mathcal{R}'$ est présenté dans la figure 6.8, la sous-hiérarchie de Galois $\mathcal{H}(\mathcal{R}')$ dans la figure 6.9.

• Si on retire ensuite $(b, 1)$ de $\mathcal{R}'$, on obtient une nouvelle relation $\mathcal{R}''$ (voir l'exemple 5.4.3, page 128). Le treillis des concepts $\mathcal{L}(\mathcal{R}'')$ associé à $\mathcal{R}'$ est présenté dans la figure 6.10, la sous-hiérarchie de Galois $\mathcal{H}(\mathcal{R}'')$ dans la figure 6.11.
◇

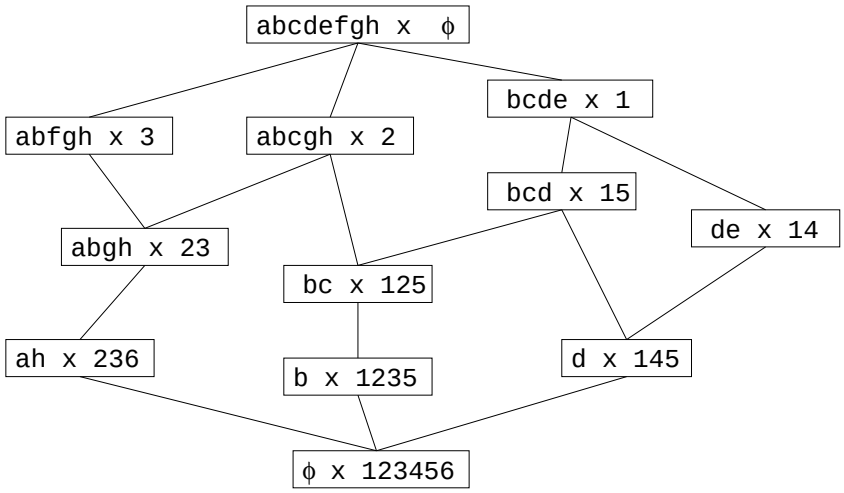


FIG. 6.8 – Treillis des concepts $\mathcal{L}(\mathcal{R}')$ de l'exemple 6.1.13.

Reconstituer l'ordre de domination

Les informations de domination permettent de reconstituer la sous-hiérarchie sur les intensions. En effet, d'après le théorème 6.1.11, il suffit de calculer l'ordre de domination

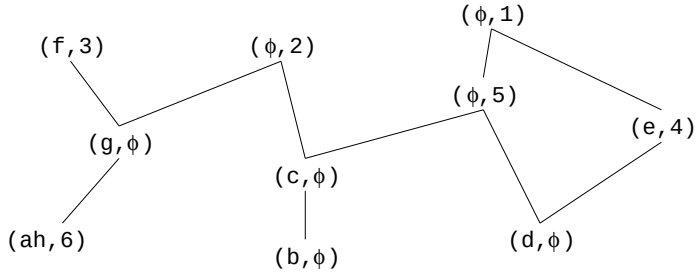


FIG. 6.9 – Sous-hiérarchie de Galois $\mathcal{H}(\mathcal{R}')$ de l'exemple 6.1.13.

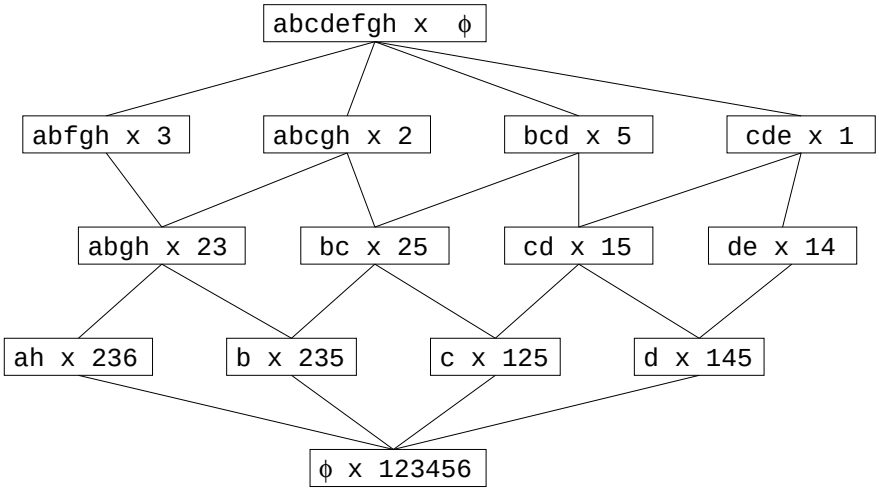


FIG. 6.10 – Treillis des concepts $\mathcal{L}(\mathcal{R}'')$ de l'exemple 6.1.13.

sur les propriétés. Cela peut se faire récursivement en déterminant dans $G_{\mathcal{R}}$ les maxmods $\tilde{m}$ non dominants du graphe et en calculant pour chacun l'ensemble P des sommets qu'il domine, puis en traitant le sous-graphe $G_{\mathcal{R}}(P \cup \mathscr{O})$. L'algorithme `ORDOM` ci-dessous réalise ce travail ; il est appelé par `ORDOM`($\emptyset$,$\mathscr{P}$). Cet algorithme utilise la table de domination L_{dom} et le graphe $G_{\mathcal{R}}$, en variables globales, et un ensemble $\mathcal{A}$ initialisé à $\emptyset$. Après le dernier appel récursif, $\mathcal{A}$ contient les arcs de la couverture de l'ordre de domination sur les propriétés de G. Ce procédé s'apparente au calcul d'une S-séquence d'un graphe orienté sans cycle.

Algorithme `ORDOM` :

Donnée : Le maxmod $\tilde{m}$ (ayant engendré l'arc $\tilde{m} \rightarrow X$, si $\tilde{m}=\emptyset$),
l'ensemble X des sommets strictement dominés par $\tilde{m}$.
Résultat : L'ordre de domination sur l'ensemble de propriétés $\tilde{m}+X$.
Début
Déterminer l'ensemble ND des maxmods non dominés de $G_{\mathcal{R}}(X \cup \mathscr{O})$;
Pour chaque $\tilde{x}$ dans ND **faire :**
  Si $\tilde{m} \neq \emptyset$ alors $\mathcal{A} \leftarrow \mathcal{A}+\{\tilde{m}\tilde{x}\}$;
  Calculer l'ensemble P des sommets strictement dominés par $\tilde{x}$;
  si $P \neq \emptyset$ alors `ORDOM`($\tilde{x}$,P) ;
End.

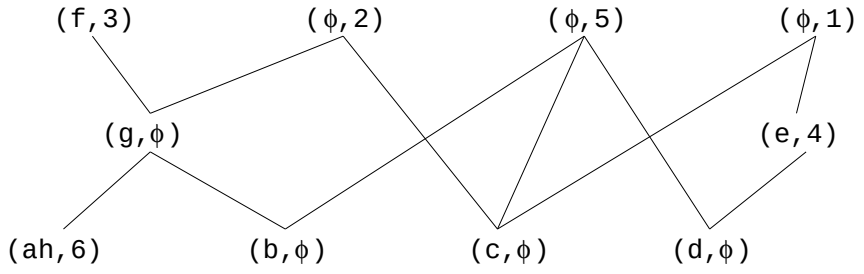


FIG. 6.11 – Sous-hiérarchie de Galois $\mathcal{H}(\mathcal{R}'')$ de l'exemple 6.1.13.

L'algorithme est validé par le double invariant suivant :

Invariant 6.1.14
- *$\tilde{m}$ domine tous les éléments de X.*
- *Un élément de ND ne peut dominer un élément de $X-ND$.*

Ainsi, les éléments de ND forment la couverture de $\tilde{m}$ dans l'ordre de domination, chaque appel récursif de `ORDOM` correspond à la construction d'un arc de l'ordre de domination.

La table L_{dom} permet de calculer en $O(m)$ l'ensemble des maxmods non dominés du graphe ; P est calculable en $O(n)$. Chaque appel de `ORDOM` est donc réalisé en $O(m)$. La complexité globale est donc en $O(nm)$. Cependant, en utilisant une partition ordonnée des maxmods de $G_{\mathcal{R}}$ avec les maxmods non dominés en début de partition, on pourrait obtenir une meilleure complexité amortie, en $O(n^2)$.

Notons que la table de domination sur les propriétés permet aussi de retrouver le concept $A \times B$ correspondant à un élément de $\mathcal{I}(\mathcal{R})$ dont l'étiquette contient une propriété x. L'intension A du concept contient exactement les propriétés que x domine : $A = \{y \in \mathscr{P} \mid \mathrm{L}_{dom}[x,y] = \emptyset\}$. L'extension B du concept est donné par : $B = \mathscr{O} - N^+(x)$, comme nous l'avons vu dans la sous-section précédente. A et B peuvent être calculés en temps proportionnel à leurs tailles (au plus $O(n)$). On peut donc ajouter le calcul du concept associé à chaque élément de $\mathcal{I}(\mathcal{R})$ à l'algorithme `ORDOM` sans augmenter sa complexité.

L_{dom}	a	b	c	d	e	f	g	h
a	$\emptyset$	$\{1\}$	$\{1,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\emptyset$	$\emptyset$
b	$\{6\}$	$\emptyset$	$\{5\}$	$\{4,5\}$	$\{4\}$	$\emptyset$	$\emptyset$	$\{6\}$
c	$\{3,6\}$	$\{3\}$	$\emptyset$	$\{4\}$	$\{4\}$	$\{3\}$	$\{3\}$	$\{3,6\}$
d	$\{2,3,6\}$	$\{2,3\}$	$\{2\}$	$\emptyset$	$\emptyset$	$\{3\}$	$\{2,3\}$	$\{2,3,6\}$
e	$\{2,3,6\}$	$\{2,3\}$	$\{2,5\}$	$\{5\}$	$\emptyset$	$\{3\}$	$\{2,3\}$	$\{2,3,6\}$
f	$\{2,6\}$	$\{1,2\}$	$\{1,2,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\{2\}$	$\{2,6\}$
g	$\{6\}$	$\{1\}$	$\{1,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\emptyset$	$\{6\}$
h	$\emptyset$	$\{1\}$	$\{1,5\}$	$\{1,4,5\}$	$\{1,4\}$	$\emptyset$	$\emptyset$	$\emptyset$

FIG. 6.12 – Table de domination sur les propriétés L_{dom} de l'exemple et 6.1.15.

Exemple 6.1.15
Exécutons l'algorithme `ORDOM` sur le graphe $G_{\mathcal{R}}$ de l'exemple 6.1.5 (page 132, la table de domination sur les propriétés L_{dom} est rappelée dans la figure 6.12).

Etape 1 :
L'appel initial est `ORDOM`($\emptyset$, $\mathscr{P}$), avec $\mathcal{A} = \emptyset$.
L'ensemble des maxmods de $G_{\mathcal{R}}$ est $\{ah, b, c, d, e, f, g\}$.
ND=$\{f, e, c\}$.
- On traite f qui est non dominé dans $G_{\mathcal{R}}$ car $\mathrm{L}_{dom}[x, f] = \emptyset$ seulement pour $x = f$. $\mathrm{L}_{dom}[f, y] = \emptyset$ pour $y \in \{a, b, g, h\}$, ce qui nous donne l'ensemble P des sommets dominés par f. On appelle `ORDOM`(f,$\{a, b, g, h\}$).
- On traite e. $P = \{d\}$, on appelle `ORDOM`(e,$\{d\}$).
- On traite c. $P = \emptyset$, pas d'appel récursif.

Etape 2 :
L'étape 1 appelle récursivement `ORDOM`(f,$\{a, b, g, h\}$) : ND=$\{g\}$.
- g est non dominé dans $G_{\mathcal{R}}(\{a, b, g, h\})$, on crée l'arc $f \rightarrow g$, $P = \{a, b, h\}$, on appelle `ORDOM`(g,$\{a, b, h\}$).

Etape 3 : L'étape 2 appelle récursivement `ORDOM`(g,$\{a, b, h\}$) : ND=$\{ah, g\}$.
- ah est non dominé dans $G_{\mathcal{R}}(\{a, b, h\})$, on crée l'arc $g \rightarrow ah$, pas d'appel récursif.
- b est non dominé dans $G_{\mathcal{R}}(\{a, b, h\})$, on crée l'arc $g \rightarrow b$, pas d'appel récursif.

Etape 4 :
L'étape 1 appelle récursivement `ORDOM`(e,$\{d\}$) : ND=$\{d\}$.
- d est non dominé dans $G_{\mathcal{R}}(\{d\})$, $P = \emptyset$, on crée l'arc $e \rightarrow d$, pas d'appel récursif.

La pile de récursivité est vide et l'algorithme prend fin.

On obtient finalement : $\mathcal{A} = \{f \rightarrow g, g \rightarrow ah, g \rightarrow b, e \rightarrow d\}$.
$\diamond$

Reconstituer la sous-hiérarchie de Galois

Les informations de domination permettent aussi de reconstituer la sous-hiérarchie de Galois $\mathcal{H}(\mathcal{R})$. On peut, dans un premier temps, construire la sous-hiérarchie des intensions $\mathcal{I}(\mathcal{R})$, à partir de la table de domination sur les propriétés $\mathrm{L}_{\mathscr{P}}$. Dans un deuxième temps on construit la sous-hiérarchie des extensions $\mathcal{E}(\mathcal{R})$ avec la table de domination sur les objets $\mathrm{L}_{\mathscr{O}}$. C'est au cours de cette deuxième phase que s'effectue la recomposition de $\mathcal{H}(\mathcal{R})$, en déterminant les éléments qui appartiennent à la fois à $\mathcal{I}(\mathcal{R})$ et à $\mathcal{E}(\mathcal{R})$. Sur ce principe, nous proposons un algorithme qui reconstitue les éléments de $\mathcal{H}(\mathcal{R})$.

Algorithme `SHG` :

Donnée : Un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$, son graphe sous-jacent $G_{\mathcal{R}}$, les tables de domination $\mathrm{L}_{\mathscr{P}}$ sur les propriétés et $\mathrm{L}_{\mathscr{O}}$ sur les objets.
Résultat : L'ensemble Elements des éléments de la sous-hiérarchie de Galois.
Initialisation :
  $\mathscr{E} \leftarrow \emptyset$;
  Atteint $\leftarrow \emptyset$;
  Elements $\leftarrow \emptyset$;
  intensions $\leftarrow \emptyset$;
Begin

// **1.** *Construire $\mathcal{I}(\mathcal{R})$ avec les intensions*
Tant que ATTEINT$\neq \mathscr{P}$ faire :
  Choisir x dans $\mathscr{P}-$ATTEINT ;
  $A \leftarrow \{y \in \mathscr{P} | \mathrm{L}_{\mathscr{P}}[x,y] = \emptyset\}$; // *propriétés dominées par x*
  $X \leftarrow \{y \in A | \mathrm{L}_{\mathscr{P}}[y,x] = \emptyset\}$; // *maxmod de x*
  ATTEINT $\leftarrow$ ATTEINT $\cup X$;
  INTENSIONS $\leftarrow$ INTENSIONS $\cup \{(A, X, \emptyset)\}$;
  ELEMENTS $\leftarrow$ ELEMENTS $\cup \{(X, \emptyset)\}$;
  // *Si besoin, calculer l'extension $B \leftarrow \mathscr{O}-N^+(x)$*
  // *Si besoin, mémoriser les arcs entre $(X, \emptyset)$ et les autre éléments de* ELEMENTS
// **2.** *Construire $\mathcal{E}(\mathcal{R})$ avec les extensions et construire $\mathcal{H}(\mathcal{R})$*
ATTEINT $\leftarrow \emptyset$;
Tant que ATTEINT$\neq \mathscr{O}$ faire :
  Choisir i dans $\mathscr{O}-$ATTEINT ;
  $B \leftarrow \{j \in \mathscr{O} | \mathrm{L}_{\mathscr{O}}[i,j] = \emptyset\}$; // *objets dominés par i*
  $Y \leftarrow \{j \in B | \mathrm{L}_{\mathscr{O}}[j,i] = \emptyset\}$; // *maxmod de i*
  ATTEINT $\leftarrow$ ATTEINT $\cup Y$;
  $A \leftarrow \mathscr{P}-N^+(i)$; // *Calcul de l'intension*
  Si A se trouve dans un élément $(A, X, \emptyset)$ de INTENSIONS
    alors // *$(X, \emptyset)$ a dèjà été calculé pour $\mathcal{I}(\mathcal{R})$*
      remplacer $(X, \emptyset)$ par (X, Y) dans ELEMENTS ;
      remplacer $(A, X, \emptyset)$ par (A, X, Y) dans INTENSIONS
    sinon // *l'élément est absent de $\mathcal{I}(\mathcal{R})$*
      ELEMENTS $\leftarrow$ ELEMENTS $\cup (\emptyset, Y)$;
      // *Si besoin, mémoriser les arcs entre $(\emptyset, Y)$ et les autre éléments*
End.

Avec $O(n)$ opérations par boucle Tant-que, l'algorithme `SHG` a une complexité en $O(n^2)$ si l'on ne tient pas compte du calcul des arcs de $\mathcal{H}(\mathcal{R})$. Pour calculer les arcs de $\mathcal{H}(\mathcal{R})$ en même temps que les éléments, on peut reprendre le principe de l'algorithme `ORDOM` que l'on adaptera pour les objets afin de prendre en compte non seulement les éléments communs à $\mathcal{I}(\mathcal{R})$ et $\mathcal{E}(\mathcal{R})$, mais aussi les arcs de la couverture de $\mathcal{H}(\mathcal{R})$ reliant un élément de $\mathcal{I}(\mathcal{R})$ à un élément de $\mathcal{E}(\mathcal{R})$.

Ainsi, notre travail sur la domination nous a permis de proposer le premier algorithme existant de maintien d'une sous-hiérarchie de Galois.

6.2 Génération du treillis des concepts

Le second problème que nous avons étudié est la génération des concepts ou d'une partie des concepts (génération des *itemsets* fermés fréquents, par exemple), que l'on construise ou non les arcs du diagramme de Hasse.

La popularité croissante du treillis de Galois, en tant qu'outil de Data Mining, a mis à l'ordre du jour le problème d'engendrer le plus efficacement possible les concepts. La finesse des algorithmes est d'autant plus cruciale que le nombre de concepts définis est

grand, comme c'est le cas pour la plupart des relations utilisées. La masse de travaux récents sur ce problème en témoigne (voir [Che69], [Gue93], [Bor86], [Nor78], [Gan84], [NR99], [KO02]).

Les algorithmes de génération peuvent revêtir plusieurs aspects : on peut vouloir engendrer et stocker tous les concepts (voir [NR99]), ou simplement atteindre chacun d'eux au moins une fois (voir [Gan84]), ou encore calculer les concepts et leur structuration, c'est-à-dire les arcs du treillis (voir [Bor86]).

6.2.1 Algorithmes existants

Pour engendrer brutalement les concepts, on peut examiner chaque partie de $\mathscr{P} \times \mathscr{O}$ en vérifiant pour chacune d'elle si elle correspond à un concept. Malgré une complexité effroyable, cette démarche semble s'être avérée intéressante dans certaines situations ; en effet, la simplicité de l'algorithme permet de concevoir un programme qui tient en quelques lignes et peut être implémenté en parallèle à un très bas niveau (utilisation par [Pas00] pour la génération des *itemsets* fréquents). Néanmoins, en raison du nombre exponentiel de parties, il n'est raisonnable d'utiliser cette technique que pour des relations de petite taille.

Pour les relations plus grosses, il est important d'engendrer les concepts sans atteindre toutes les parties. Ce problème a donné lieu à de nombreuses publications dans les trente dernières années et les algorithmes se sont affinés progressivement.

Un des premiers algorithmes publiés fut celui de Chein ([Che69]) ; cet algorithme engendre le treillis par couches successives, en définissant des candidats possibles par combinaisons de concepts de la couche précédente ; il a une complexité en pire cas exponentielle par concept engendré (voir [Gue93]).

L'algorithme de Bordat ([Bor86]) a apporté une amélioration, puisque il tourne en $O(n^3)$ par concept, en se basant sur un parcours en largeur du treillis ; une des particularités intéressantes de cet algorithme est qu'il calcule aussi les arcs du diagramme de Hasse du treillis. Cette complexité en temps a été récemment améliorée par Nourine et Raynaud ([NR99]) à $O(n^2)$ par concept.

Tous ces algorithmes nécessitent un espace mémoire exponentiel et stockent les concepts calculés.

Quand on n'a pas besoin de stocker les concepts, mais seulement de les atteindre au moins une fois chacun, le problème d'espace mémoire devient plus simple, bien que le temps de traitement par concept soit paradoxalement plus élevé : le meilleur algorithme de ce type est dû à Ganter ([Gan84]) et tourne en $O(n^3)$ par concept, en utilisant la notion intéressante "d'ordre lectique" (*lectic order*), qui évite d'explorer toutes les parties, mais pas de calculer le même concept $O(n)$ fois.

Pour une étude plus détaillée des algorithmes de construction de treillis des concepts, le lecteur peut se reporter au panorama de Guénoche [Gue93], malheureusement antérieur à la publication de [NR99], et au papier très récent de Sergei (voir [KO02]) qui propose une

analyse à la fois expérimentale et théorique de ces principaux algorithmes de génération.

6.2.2 Génération des concepts par calcul des séparateurs minimaux

Pour illustrer l'usage qu'on peut faire de notre nouveau paradigme présenté au chapitre 4, nous allons utiliser les travaux récents réalisés pour engendrer tous les séparateurs minimaux ou les xy-séparateurs minimaux d'un graphe (voir [SL97], [KK98], [BBC00], [She99]) pour montrer que nous pouvons facilement égaler les meilleurs algorithmes existants pour engendrer et stocker tous les concepts.

Pour cela nous allons construire un nouveau graphe intermédiaire $H_{\mathcal{R}}$ en ajoutant à notre graphe sous-jacent $G_{\mathcal{R}}$ tel qu'il est décrit dans le chapitre 4 deux sommets simpliciaux x et y tels que $N(x) = \mathscr{P}$ et $N(y) = \mathscr{O}$.

Propriété 6.2.1
Soit $\mathscr{S}(G_{\mathcal{R}})$ l'ensemble des xy-séparateurs minimaux de $G_{\mathcal{R}}$. L'ensemble des xy-séparateurs minimaux de $H_{\mathcal{R}}$ est exactement $\{\mathscr{P}\} \cup \{\mathscr{O}\} \cup \mathscr{S}(G_{\mathcal{R}})$.

Cette simple remarque permet d'utiliser le travail de Shen [She99] sur la génération des xy-séparateurs minimaux d'un graphe : [She99] revendique une complexité en $O(n^2)$ par xy-séparateur minimal pour les engendrer et les stocker tous. Cette complexité est aussi bonne que celle de [NR99]. Remarquons que [She99] fait une analyse en moyenne de la place requise qui paraît très intéressante.

Notons pour terminer que l'énumération des xy-séparateurs minimaux d'un graphe pourrait peut-être descendre à $O(m)$ par séparateur minimal, ce qui conduirait directement à une amélioration correspondante de la génération des concepts.

6.2.3 Génération des concepts par le calcul de la couverture

Nous allons maintenant décrire un principe algorithmique qui nous permettra d'améliorer le problème de la génération sans stockage des concepts. Pour le même coût, nous engendrons les arcs du treillis. Ce principe requiert un espace mémoire polynomial ($O(nm)$).

Nous proposerons deux implémentations de ce principe. Un premier algorithme, d'implémentation très simple, tourne en $O(nm)$ par concept, ce qui correspond déjà aux algorithmes ayant le meilleur temps pour engendrer les éléments sans les stocker (algorithme de [Gan84]) mais aussi pour engendrer la structure (algorithme de [Bor86]).

Le même principe algorithmique nous fournit un deuxième algorithme qui tourne en temps $O(nm)$ par chaîne maximale parcourue. Ceci améliore [NR99] puisque nous avons une complexité en temps comparable sans avoir besoin d'un espace mémoire exponentiel ; ceci améliore aussi significativement la complexité en $O(n^3)$ par concept de Ganter. De

plus, le nombre de concepts tend à devenir exponentiel quand la relation est dense ; dans ce cas, $m \in O(n)$ et notre complexité tombe à $O(n^2)$ par chaîne maximale parcourue.

Le principe algorithmique que nous introduisons ici présente des similitudes avec celui de [Bor86]. Il atteint récursivement chaque concept $A \times B$ et calcule sa couverture, en travaillant à chaque étape sur la sous-relation $\mathcal{R}((\mathscr{P}-A), B)$.

Une des améliorations apportées est que chaque concept hérite des informations transmises par les concepts précédemment calculés, afin d'éviter d'engendrer un même concept plus d'une fois. Ceci constitue une avancée dans la génération des concepts.

Une autre amélioration est que nous utilisons un parcours en profondeur du treillis, ce qui nous autorise à ne stocker qu'un nombre polynomial d'octets, puisque qu'un treillis des concepts, bien que de taille potentiellement exponentielle, est de faible hauteur ($O(n)$).

Enfin, la façon dont nous calculons la couverture d'un concept est foncièrement nouvelle, puisque nous utilisons nos résultats sur la domination entre maxmods.

Nous allons tout d'abord décrire notre procédé algorithmique général. Nous présenterons ensuite les deux algorithmes, qui se distinguent par la façon de calculer les maxmods non dominants ; le premier est très simple mais la complexité dans le pire cas n'est guère meilleure que celle de Ganter, le second, plus sophistiqué, utilise la table de cardinalité présentée au chapitre 5, de façon à gérer efficacement la mise à jour des informations de domination.

A. Principe algorithmique

Nous avons vu au chapitre 5 que l'on peut utiliser la domination pour calculer la couverture d'un concept (théorème 5.3.6), nous allons utiliser ce théorème pour calculer récursivement tous les concepts, en commençant par le bottom. Si on ne prend pas de précautions, ce procédé engendrera chaque concept exactement autant de fois qu'il a de prédécesseurs.

Notre but est d'engendrer chaque concept exactement une fois. Comme l'ordre sur les concepts est défini par inclusion sur les intensions, tout concept $A' \times B'$ qui est un descendant de $A \times B$ vérifie $A \subseteq A'$. Puisque notre algorithme utilise une approche en profondeur, quand un maxmod X est utilisé pour engendrer un concept $(A+X) \times B'$ alors tous les concepts ayant X dans leur intension serons définis par l'appel récursif sur $(A+X) \times B'$. Si un concept frère de $(A+X) \times B'$ utilise un maxmod contenant un sommet x de X alors ce concept aura déjà été engendré précédemment. Si nous stockons les informations sur les maxmods qui ont déjà été utilisé par un frère ou le frère d'un ancêtre, nous évitons de calculer le même concept plus d'une fois.

Procédé algorithmique 6.2.2 sur un concept $A \times B$.

CALCULER l'ensemble ND des maxmods non dominants de $G_{\mathcal{R}}((\mathscr{P}-A) \cup B)$;
// *Chaque maxmod X de* ND *définit un concept couvrant de $A \times B$.*
CALCULER l'ensemble Nouveau des maxmods de ND sans sommet déjà traité ;
// *Chaque maxmod X de* Nouveau *définit un nouveau concept couvrant $A \times B$.*
Pour chaque maxmod X de Nouveau **faire :**
  APPLIQUER récursivement le processus au nouveau concept d'intension $A+X$;

AJOUTER tous les sommets de X à l'ensemble des sommets déjà traités.

Le problème de complexité de ce procédé peut provenir du calcul de l'ensemble des maxmods non-dominants et l'analyse de complexité en pire cas dépendra lourdement de la manière dont ce problème est résolu.

B. Un premier algorithme basé sur le calcul direct des maxmods

Calcul des maxmods :

La manière la plus directe de calculer les maxmods non-dominants est d'utiliser le procédé algorithmique 5.3.8 : répétitivement trouver un maxmod X de degré minimal, calculer les maxmods qui le dominent, enlever ces maxmods ainsi que X de l'ensemble des sommets avant de recommencer. Nous nous appuyons sur l'algorithme d'affinement de partitions de [HsM99], basé sur l'algorithme `Lex-BFS`, pour partitionner les sommets de $G_{\mathcal{R}}$ en maxmods. On remarquera que ces algorithmes permettent d'obtenir une partition ordonnée des sommets en maxmods, dans laquelle un maxmod est obligatoirement placé après tous les maxmods qu'il domine ; le premier maxmod de la partition ordonnée est ainsi non dominant. Nous en déduisons un algorithme `POND` qui calcule une partition ordonnée des propriétés de $G_{\mathcal{R}}$ en maxmods, avec une complexité linéaire ($O(m)$).

Algorithme `POND` :
Donnée : Le graphe sous-jacent $G_{\mathcal{R}}$ à un contexte $(\mathscr{P}, \mathscr{O}, \mathcal{R})$.
Résultat : Une partition ordonnée PO des sommets de $\mathscr{P}$ en maxmods de $G_{\mathcal{R}}$.
Initialisation : PO $\leftarrow (\mathscr{P})$; // *Une seule classe $\mathscr{P}$ au départ.*
Début
Pour chaque objet y de $\mathscr{O}$ **faire :**
  Pour chaque classe K de PO telle que $|K| > 1$ **faire :**
    // *Partitionner la classe K en K_1 et K_2.*
    $K_1 = K - N^+(y)$;
    $K_2 = K \cap N^+(y)$;
    Si K_1 et K_2 sont toutes deux non vides **alors**
      Remplacer, dans PO, K par K_1 suivi de K_2 ;
Fin.

Invariant 6.2.3
Après chaque partition d'une classe K en K_1 et K_2 dans l'algorithme `POND`*, aucun sommet de K_1 ne domine un sommet de K_2.*

Preuve : Soient $x_1 \in K_1$ et $x_2 \in K_2$. x_1y n'est pas une arête de $G_{\mathcal{R}}$ puisque $K_1 = K - N^+(y)$ et donc $x_1 \notin N^+(y)$; par contre x_2y est une arête de $G_{\mathcal{R}}$ puisque $K_2 = K \cap N^+(y)$ et donc $x_2 \in N^+(y)$. Par conséquent x_1 ne peut dominer x_2.
◇

On peut en déduire que, si K_i et K_j sont deux classes obtenues par une exécution de `POND`, avec K_i avant K_j, alors aucun sommet de K_i ne domine un sommet de K_j. En

particulier, on est assuré que la première classe est non dominante ; elle correspond aux moplex définis par une exécution de `Lex-BFS` (voir [BB98]).

Exemple 6.2.4
Exécutons l'algorithme `POND` sur le graphe de l'exemple 6.1.5 (page 132), dont l'ordre de domination a été présenté dans la figure 6.5 (page 136).

Initialisation : PO$=(\{a,b,c,d,e,f,g,h\})$.

Objet 1 **:** $N^+(1)=\{a,f,g,h\}$;
$K=\{a,b,c,d,e,f,g,h\}$, $K_1=\{b,c,d,e\}$ et $K_2=\{a,f,g,h\}$,
PO$=(\{b,c,d,e\},\{a,f,g,h\})$.

Objet 2 **:** $N^+(2)=\{d,e,f\}$;
$K=\{b,c,d,e\}$, $K_1=\{b,c\}$ et $K_2=\{d,e\}$, PO$=(\{b,c\},\{d,e\},\{a,f,g,h\})$;
$K=\{a,f,g,h\}$, $K_1=\{a,g,h\}$ et $K_2=\{f\}$, PO$=(\{b,c\},\{d,e\},\{a,g,h\},\{f\})$.

Objet 3 **:** $N^+(3)=\{c,d,e\}$, PO$=(\{b\},\{c\},\{d,e\},\{a,g,h\},\{f\})$.

Objet 4 **:** $N^+(4)=\{a,b,c,f,g,h\}$, PO inchangé.

Objet 5 **:** $N^+(5)=\{a,b,e,f,g,h\}$, PO$=(\{b\},\{c\},\{d\},\{e\},\{a,g,h\},\{f\})$.

Objet 6 **:** $N^+(6)=\{b,c,d,e,f,g\}$, PO$=(\{b\},\{c\},\{d\},\{e\},\{a,h\},\{g\},\{f\})$.

On obtient ainsi une partition ordonnée des maxmods de $G_{\mathcal{R}}$:
$(\{b\},\{c\},\{d\},\{e\},\{a,h\},\{g\},\{f\})$.
b, c et d sont des maxmods non dominants ; e n'est dominé que par d qui vient avant ; ah est non dominant ; g ne domine que ah et b qui viennent avant ; f, qui domine g, ah et b vient en dernier.
◇

L'algorithme :

Nous donnons maintenant un algorithme récursif correspondant à cette façon de calculer les maxmods :

Algorithme `CONCEPTS-1` :

Donnée : Un concept $A\times B$ du contexte $(\mathscr{P},\mathscr{O},\mathcal{R})$, un ensemble ATTEINT.
Résultat : Tous les éléments de la couverture de $A\times B$
Début :
$G\leftarrow G_{\mathcal{R}}((\mathscr{P}-A-\text{ATTEINT})\cup B)$;
// $G_{\mathcal{R}}((\mathscr{P}-A)\cup B)$ *est le graphe sous-jacent au contexte* $(\mathscr{P}-A,B,\mathcal{R}(\mathscr{P}-A,B))$.
PO $\leftarrow$ `POND`(G) ;
Tant que PO $\neq\emptyset$ **faire :**
  Prendre le premier maxmod X de PO ;
  Si $X\cap$ATTEINT$=\emptyset$ **alors**
    // X *engendre le concept* $A+X\times B-N^+(X)$.
  `CONCEPTS-1`$(A+X\times B-N^+(X)$, `Atteint`$)$;
  Supprimer, de PO, X ainsi que tous les maxmods qui le dominent ;
  Ajouter, à ATTEINT, X ainsi que tous les maxmods qui le dominent ;
Fin.

L'algorithme est initialement appelé par `CONCEPTS-1`($\emptyset \times \mathscr{O}$), avec ATTEINT$= \emptyset$. Chaque exécution de `CONCEPTS-1` réalise un appel à `POND` sur le sous-graphe de $G_{\mathcal{R}}$ correspondant au concept dont on calcule la couverture.

Analyse de complexité :

• Complexité en temps : chaque étape de l'algorithme `CONCEPTS-1` réalise un appel à l'algorithme `POND`, en $O(m)$. Rechercher et supprimer dans PO tous les maxmods qui dominent le maxmod en cours de traitement coûte $O(m)$; cette recherche est faite pour chaque concept de la couverture qui est engendré. Cette recherche coûte $O(km)$ par concept, où k est le degré sortant du concept (notons que $k \leq n$), ce qui nous donne globalement $O(\Gamma m)$ pour le parcours complet du treillis, où Γ est le nombre d'arcs du diagramme de Hasse.

• Complexité en espace : la pile de récursivité contient au plus $O(n)$ concepts de taille $O(n)$ chacun, pour conserver les maxmods de PO qui restent à traiter ; la complexité spatiale est donc en $O(n^2)$.

Exemple 6.2.5
Exécutons l'algorithme `CONCEPT-1` sur l'exemple 6.1.5 (page 132). La figure 6.13 illustre cette exécution.
L'appel initial se fait sur le concept bottom $\emptyset \times 123456$, avec $G = G_{\mathcal{R}}$ et ATTEINT$= \emptyset$.

Etape 1 : `POND`($G_{\mathcal{R}}$) donne : PO$= (\{b\}, \{c\}, \{d\}, \{e\}, \{a, h\}, \{g\}, \{f\})$.
On choisit le premier maxmod de PO : $X = b$, $N^+(b) = \{4, 5, 6\}$, $A + X = \emptyset + \{b\} = \{b\}$ et $B - N^+(b) = \{1, 2, 3\}$, le concept $b \times 123$ est engendré. Dans $G = G_{\mathcal{R}}$, f et g dominent b, on les retire de PO en même temps que b. PO devient $(\{c\}, \{d\}, \{e\}, \{a, h\})$, ATTEINT devient $\{b, f, g\}$.
On choisit $c \notin$ ATTEINT : $N^+(c) = \{346\}$, $A + X = c$ et $B = 125$, le concept $c \times 125$ est engendré. c n'est pas dominé, PO devient $(\{d\}, \{e\}, \{a, h\})$ et ATTEINT $\{b, c, f, g, h\}$.
On choisit d : $N^+(d) = \{2, 3, 6\}$, le concept $d \times 145$ est engendré. e domine d, PO devient $(\{a, h\})$ et ATTEINT $\{b, c, d, e, f, g\}$.
On choisit ah : $N^+(ah) = \{1, 4, 5\}$, $A + X = ah$ et $B = 236$, le concept $ah \times 236$ est engendré. PO devient vide et ATTEINT$= \mathscr{P}$.

Etape 2 : L'étape 1 appelle récursivement $b \times 123$ avec ATTEINT$= \emptyset$.
On travaille dans le sous-graphe $G = G_{\mathcal{R}}(\{a, c, d, e, f, g, h, 1, 2, 3\})$ dans lequel $N^+(a) = N^+(g) = N^+(h) = \{1\}$, $N^+(c) = \{3\}$, $N^+(d) = N^+(e) = \{2, 3\}$, $N^+(f) = \{1, 2\}$.
PO$= (\{c\}, \{d, e\}, \{agh\}, \{f\})$.
c engendre $bc \times 12$; de domine c, PO devient $(\{agh\}, \{f\})$ et ATTEINT $\{c, d, e\}$.
agh engendre $abgh \times 23$; f domine agh, PO devient vide et ATTEINT$= \mathscr{P}$.

Etape 3 : L'étape 2 appelle récursivement $bc \times 12$ avec ATTEINT$= \emptyset$; dans $G = G_{\mathcal{R}}(\{a, d, e, f, g, h, 1, 2\})$, $N^+(a) = N + (g) = N^+(h) = \{1\}$, $N^+(d) = N^+(e) = \{2\}$, $N^+(f) = \{1, 2\}$.
PO$= (\{d, e\}, \{a, g, h\}, \{f\})$.
de engendre $bcde \times 1$; PO$= (\{a, g, h\}, \{f\})$, ATTEINT$= \{d, e\}$.
agh engendre $abfgh \times 3$; f domine agh, PO devient vide.

Etape 4 : L'étape 3 appelle récursivement $bcde \times 1$ avec ATTEINT$= \emptyset$;
$G = G_{\mathcal{R}}(a, f, g, h, 1)$. PO$= (\{a, f, g, h\})$.

$afgh$ engendre $\mathscr{P} \times \emptyset$; PO devient vide.

Etape 5 : L'étape 4 appelle récursivement le top $\mathscr{P} \times \mathscr{O}$ avec ATTEINT= $\emptyset$; $G = G_{\mathcal{R}}(\emptyset)$, PO est vide.
Aucun concept n'est engendré.

Etape 6 : L'étape 3 appelle récursivement $abcgh \times 2$ avec ATTEINT= $\{d, e\}$; $G = G_{\mathcal{R}}(a, b, c, f, g, h, 2)$.
PO= $(\{d, e, f\})$.
d est déjà atteint. PO devient vide.
Aucun concept n'est engendré.

Etape 7 : L'étape 2 appelle récursivement $abgh \times 23$ avec ATTEINT= $\{c, d, e\}$.
PO= $(\{f\}, \{c, d, e\})$.
f engendre $abfgh \times 3$; cde (par ailleurs déjà atteint) domine f. PO devient vide.

Etape 8 : L'étape 7 appelle récursivement $abfgh \times 3$ avec ATTEINT= $\{c, d, e\}$.
PO= $(\{c, d, e\})$.
c est déjà atteint. PO devient vide. Aucun concept n'est engendré.

Etape 9 : L'étape 1 appelle récursivement $c \times 125$ avec ATTEINT= $\{b, f, g\}$.
PO= $(\{b\}, \{d\}, \{e\}, \{agh\}, \{f\})$.
b est déjà atteint. e et agh dominent b, PO devient $(\{d\}, \{f\})$ et ATTEINT $\{a, b, e, f, g, h\}$.
d engendre $cd \times 15$; f domine agh, PO devient vide.

Etape 10 : L'étape 9 appelle récursivement $cd \times 15$ avec ATTEINT= $\{b, f, g\}$.
PO= $(\{be\}, \{afgh\})$.
b est déjà atteint. $afgh$ domine b, PO devient vide.
Aucun concept n'est engendré.

Etape 11 : L'étape 1 appelle récursivement $d \times 145$ avec ATTEINT= $\{b, c, f, g\}$.
PO= $(\{e\}, \{c\}, \{b\}, \{afgh\})$.
e engendre $de \times 14$; b et $afgh$ dominent e, PO devient $(\{c\})$, ATTEINT= $\{a, b, c, e, f, g\}$.
c est déjà atteint. PO devient vide.

Etape 12 : L'étape 11 appelle récursivement $de \times 14$ avec ATTEINT= $\{b, c, f, g\}$.
PO= $(\{b, c\}\{afgh\})$.
b est déjà atteint. $afgh$ domine bc, PO devient vide.
Aucun concept n'est engendré.

Etape 13 : L'étape 1 appelle récursivement $ah \times 236$ avec ATTEINT= $\{b, c, d, e, f, g\}$.
PO= $(\{b, g\}, \{c\}, \{d, e\})$.
b est déjà atteint. c et de dominent b, PO devient vide.
Aucun concept n'est engendré, la pile de récursivité est vide et l'algorithme prend fin.
$\diamond$

Expérimentalement, cet algorithme, implémenté comme indiqué ci-dessus, tourne rapidement grâce aux informations héritées sur les sommets déjà traités, puisqu'à chaque étape où un maxmod X est ajouté à l'ensemble des sommets déjà traités, les maxmods qui dominent X sont aussi ajoutés.

Cependant, il est possible d'améliorer le comportement dans le pire cas du calcul des concepts, en utilisant une approche plus sophistiquée pour calculer les maxmods non-dominants. C'est ce que nous proposons dans le deuxième algorithme décrit ci-dessous

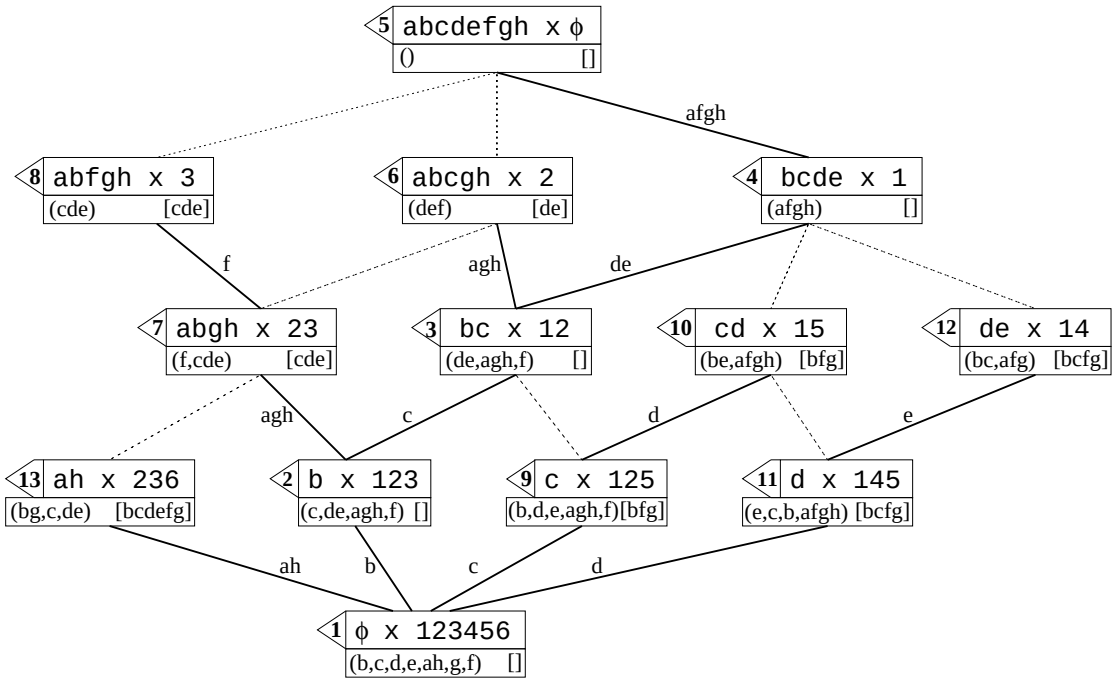


FIG. 6.13 – Le treillis des concepts $\mathcal{L}(\mathcal{R})$ de la relation $\mathcal{R}$ de l'exemple 6.1.5.

Les concepts sont numérotés dans l'ordre préfixe en suivant l'exécution récursive de l'exemple 6.2.5. La partition ordonnée PO en maxmods apparaît entre parenthèses et les ensembles hérités de sommets déjà traités entre crochets. Chaque arête de l'arbre recouvrant défini par l'exécution est étiquetée par le maxmod non dominant utilisé pour calculer le concept suivant.

qui va nous permettre de passer de $O(nm)$ par concept ($O(\Gamma m)$ pour le treillis, avec Γ nombre d'arcs dans le diagramme de Hasse du treillis) à $O(m)$ par concept (et $O(nm)$ par chaîne maximale du treillis).

C. Un algorithme utilisant la table de domination

Mise à jour des informations de domination :

Afin de répondre efficacement aux requêtes sur les maxmods non-dominants, nous utilisons la table de cardinalité sur les propriétés, présentée dans le chapitre 5 et déjà utilisée dans la première section de ce chapitre pour maintenir une sous-hiérarchie de Galois. Comme nous l'avons vu au chapitre 5, les informations de domination peuvent être héritées le long d'une chaîne maximale. Le maintien de la table de cardinalité au cours du parcours en profondeur évitera de recalculer toutes les informations de domination à chaque étape de l'algorithme.

Le mécanisme d'héritage le long d'une chaîne maximale est le suivant : quand on monte dans le treillis depuis un concept $A \times B$ qui correspond au graphe $G_{\mathcal{R}}((\mathscr{P}-A) \cup B)$, vers un autre concept $(A+X) \times B$ qui le couvre et qui correspond au graphe $G_{\mathcal{R}}((\mathscr{P}-(A \cup X)) \cup (B-N^{+}(X)))$, deux choses vont se produire :

1. L'ensemble de propriétés X disparaît ;
2. L'ensemble d'objets $N^+(X)$ disparaît.

Nous reprendrons les notations $T_{\mathscr{P}}$ et $D_{\mathscr{P}}$ de la section 5.2. Les tableaux $T_{\mathscr{P}}$ et $D_{\mathscr{P}}$ sont modifiés, à chaque étape, avant un appel récursif par un pré-traitement (*pre-update*), puis remis ensuite dans l'état originel par un post-traitement (*post-updated*). Le nombre de modifications apportées aux tableaux le long d'une chaîne maximale du treillis ne dépassera pas la taille de la table de domination, à savoir $O(nm)$, puisque la taille de la table de domination diminue le long d'une chaîne maximale du treillis.

L'algorithme :

L'algorithme est initialement appelé par `CONCEPTS-2(`$\emptyset \times \mathscr{O}$, $\emptyset$`)` sur un ensemble ATTEINT vide. Les tableaux $T_{\mathscr{P}}$ et $D_{\mathscr{P}}$ sont construits à partir de $G_{\mathcal{R}}$ selon la description donnée dans le paragraphe précédent. Prétraitement et post-traitement ont le même squelette algorithmique et pourraient donc être regroupés en un même algorithme différenciant, grâce à un paramètre, le pré-traitement du post-traitement (voir [BBS02]).

Algorithme `CONCEPTS-2` :

Données : Un concept $A \times B$, un ensemble ATTEINT $\subseteq \mathscr{P}$.
Résultat : Les successeurs directs non encore atteints de $A \times B$.
Début
Initialisation :
  $G \leftarrow G_{\mathcal{R}}((\mathscr{P}-A) \cup B)$;
  CALCULER dans G la partition de $\mathscr{P}-A$ en maxmod ;
  // *Le maxmod auquel appartient un sommet $x \in \mathscr{P}$ est noté* $\tilde{x}$.
  Pour x **dans** ATTEINT **faire :**
    ATTEINT $\leftarrow$ ATTEINT $\cup \tilde{x}$;
//*1. Calculer l'ensemble* ND *des maxmod non-dominants de G.*
ND $\leftarrow \emptyset$;
Pour x **dans** $\mathscr{P}-A$ **faire :**
  Si $D[x] = |\tilde{x}|$ **alors** ND $\leftarrow$ ND $\cup \tilde{x}$;
//*2. Au besoin, engendrer la couverture de $A \times B$.*
//**Pour** X **dans** ND **faire :**
// $A' \leftarrow A+X$;
// $B' \leftarrow \mathscr{O}-N^+(X)$;
// AFFICHER$(A' \times B')$;
//*3. Engendrer les descendants non atteints de $A \times B$.*
Pour X **dans** ND $-$ ATTEINT **faire :**
  $A' \leftarrow A+X$;
  $B' \leftarrow \mathscr{O}-N^+(X)$;
  AFFICHER$(A' \times B')$;
  // *Pour la génération de "frequent sets", tester la taille de B' ;*
    si elle est trop petite, prendre l'élément X suivant de ND–MARKED.
  `PréTraitement` ;
  `CONCEPTS-2(` $A' \times B'$, ATTEINT) ;
  `PostTraitement` ;
  ATTEINT $\leftarrow$ ATTEINT $\cup X$;
Fin.

Algorithme `PréTraitement` :

Donnée : Les valeurs courantes de X et $A \times B$ dans `CONCEPTS-2`.
Résultat : Les tableaux $T_{\mathscr{P}}$ et $D_{\mathscr{P}}$ modifiés avant l'appel récursif.
Début
Choisir un représentant x dans X ;
//1. Mettre à jour le tableau $D_{\mathscr{P}}$ en simulant le retrait des propriétés de X.
Pour y **dans** $(\mathscr{P}-A) - X$ **faire :**
  Si $T_{\mathscr{P}}[y,x] = 0$ **alors**
    $D_{\mathscr{P}}[y] \leftarrow D_{\mathscr{P}}[y]-|X|$;
//2. Mettre à jour les tableaux $T_{\mathscr{P}}$ et $D_{\mathscr{P}}$ en simulant le retrait des objets de $N^+(x)$.
Pour j **dans** $N^+(x)$ **faire :**
  $Z \leftarrow N^+(j)-X$;
  $U \leftarrow (\mathscr{P}-A) - Z - X$;
  Pour (u,z) **dans** $U \times Z$ **faire :**
    $T_{\mathscr{P}}[u,z] \leftarrow T_{\mathscr{P}}[u,z]-1$;
    Si $T_{\mathscr{P}}[u,z] = 0$ **alors** $D_{\mathscr{P}}[u] \leftarrow D_{\mathscr{P}}[u]+1$;
Fin.

Algorithme `PostTraitement` :

Donnée : Les valeurs courantes de X et $A \times B$ dans `CONCEPTS-2`.
Résultat : Les tableaux $T_{\mathscr{P}}$ et $D_{\mathscr{P}}$ remis en l'état après l'appel récursif.
Début
Choisir un représentant x dans X ;
//1. Mettre à jour le tableau $D_{\mathscr{P}}$ en simulant le retrait des propriétés de X.
Pour y **dans** $(\mathscr{P}-A) - X$ **faire :**
  Si $T_{\mathscr{P}}[y,x] = 0$ **alors**
    $D_{\mathscr{P}}[y] \leftarrow D_{\mathscr{P}}[y]+|X|$;
//2. Mettre à jour les tableaux $T_{\mathscr{P}}$ et $D_{\mathscr{P}}$ en simulant le retrait des objets de $N^+(x)$.
Pour j **dans** $N^+(x)$ **faire :**
  $Z \leftarrow N^+(j)-X$;
  $U \leftarrow (\mathcal{P}-A) - Z - X$;
  Pour (u,z) **dans** $U \times Z$ **faire :**
    $T_{\mathscr{P}}[u,z] \leftarrow T_{\mathscr{P}}[u,z]+1$;
    Si $T_{\mathscr{P}}[u,z] = 1$ **alors** $D_{\mathscr{P}}[u] \leftarrow D_{\mathscr{P}}[u]-1$;
Fin.

Analyse de la complexité :

• Complexité en temps : chaque étape de l'algorithme `CONCEPTS-2` nécessite de calculer les maxmods du graphe, ce qui peut être fait en $O(m)$. En utilisant le tableau D, trouver l'ensemble des maxmods non-dominants coûte $O(n)$. Comparer ces maxmods non-dominants avec ATTEINT coûte $O(n)$. Ainsi un concept est calculé en $O(m)$. Comme on l'a vu précédemment, la mise à jour des informations de domination par les algorithmes `PréTraitement` et `PostTraitement` coûte globalement $O(nm)$ par chaîne maximale.

• Complexité en espace : la pile de récursivité contient au plus $O(n)$ concepts de taille $O(n)$ chacun, ATTEINT a une taille en $O(n)$; $T_{\mathscr{P}}$ contient $O(nm)$ bits et $D_{\mathscr{P}}$ contient

$O(n)$ sommets ; la complexité spatiale est donc en $O(nm)$.

Exemple 6.2.6
Reprenons l'exemple 6.1.5 (page 132) pour exécuter l'algorithme `CONCEPTS-2`. La figure 6.14 illustre cette exécution.

Les tables $T_{\mathscr{P}}$ et $D_{\mathscr{P}}$ sont celles de l'exemple 5.4.5 (page 130) :

$T_{\mathscr{P}}$	a	b	c	d	e	f	g	h
a	0	1	2	3	2	0	0	0
b	1	0	1	2	1	0	0	1
c	2	1	0	1	1	1	1	2
d	3	2	1	0	0	1	2	3
e	3	2	2	1	0	1	2	3
f	2	2	3	3	2	0	1	2
g	1	1	2	3	2	0	0	1
h	0	1	2	3	2	0	0	0

$D_{\mathscr{P}}$	a	b	c	d	e	f	g	h
	2	1	1	1	2	5	4	2

Etape 1 : L'exécution commence avec le bottom $\emptyset \times 123456$. Dans $G = G_{\mathcal{R}}$, les maxmods non dominants sont $\{a, h\}$, $\{b\}$, $\{c\}$ et $\{d\}$. La couverture de $\emptyset \times 123456$ est : $ah \times 236$, $b \times 123$, $c \times 125$, $d \times 145$. L'ensemble ATTEINT des sommets déjà traités est vide. On choisit de traiter le concept $ah \times 236$.

Etape 2 : On traite le concept $ah \times 236$. T et D sont prétraitées en conséquence : puisque les objets 1, 4 et 5 disparaissent, chacun des couples (x, y) issus des différents produits cartésiens $\{b, c, d, e\} \times \{f, g\}$, $\{d, e\} \times \{b, c, f, g\}$ et $\{c, d\} \times \{b, e, f, g\}$ va décrémenter d'une unité la valeur de la case correspondante $T[x, y]$; D est mis à jour en conséquence. On obtient les nouvelles tables T et D suivantes :

$T_{\mathscr{P}}$	b	c	d	e	f	g
b	0	0	0	0	0	0
c	1	0	0	0	1	1
d	2	1	0	0	1	2
e	2	1	0	0	1	2
f	1	1	0	0	0	1
g	0	0	0	0	0	0

$D_{\mathscr{P}}$	b	c	d	e	f	g
	2	3	6	6	3	2

Le graphe G devient $G_{\mathcal{R}}(\{b, c, d, e, f, g, 2, 3, 6\})$. Les maxmods de G sont : $\{b, g\}$, $\{c\}$, $\{d, e\}$, $\{f\}$; parmi eux, seul $\{b, g\}$ est non dominant. On engendre le concept $abgh \times 23$.

Etape 3 : On traite $abgh \times 23$. Les maxmods non dominants sont : $\{c\}$ et $\{f\}$. On engendre les concepts $abcgh \times 2$ et $abfgh \times 3$. $abfgh \times 3$ est choisi pour être traité ensuite.

Etape 4 : On traite $abfgh \times 3$. Le seul maxmod non dominant est : $\{c, d, e\}$. On engendre l'élément maximum $abcdefgh \times \emptyset$.

Etape 5 : On traite $abcdefgh \times \emptyset$. Le graphe G obtenu est vide. On ne peut engendrer aucun nouveau concept.

Etape 6 : L'étape 3 appelle récursivement $abcgh \times 2$, avec ATTEINT$=\{f\}$. Les maxmods non dominants sont : $\{d, e, f\}$; puisque f est dans ATTEINT, aucun nouveau concept n'est engendré.

Etape 7 : L'étape 1 appelle récursivement $c \times 125$ avec ATTEINT$=\{a, h\}$. Les maxmods non dominants sont : $\{b\}$ et $\{d\}$. On engendre les concepts $bc \times 12$ et $cd \times 15$. $bc \times 12$ est choisi pour être traité ensuite.

Etape 8 : On traite $bc \times 12$, avec ATTEINT$=\{a, h\}$. Les maxmods non dominants sont : $\{d, e\}$ et $\{a, g, h\}$. Puisque a et h sont dans ATTEINT, on ne peut utiliser que $\{d, e\}$ pour engendrer un nouveau concept : $bcde \times 1$.

Etape 9 : On traite ensuite $bcde \times 1$, avec ATTEINT$=\{a, h\}$. Le seul maxmod non dominant est : $\{a, f, g, h\}$. Puisque a et h sont dans ATTEINT, on ne peut engendrer aucun nouveau concept.

Etape 10 : L'étape 7 appelle récursivement $cd \times 15$, avec ATTEINT$=\{a, b, h\}$; $\{a, h\}$ est hérité du concept $ah \times 236$, un frère du concept père $c \times 125$; $\{b\}$ est hérité du concept frère $bc \times 12$. Le seul maxmod non dominant est : $\{b, e\}$. Puisque b est dans ATTEINT, on ne peut engendrer aucun nouveau concept.

Etape 11 : L'étape 1 appelle récursivement $b \times 123$, avec ATTEINT$=\{a, c, h\}$. Les maxmods non dominants sont : $\{a, g, h\}$ et $\{c\}$. Puisque a, c et h sont dans ATTEINT, on ne peut engendrer aucun nouveau concept.

Etape 12 : L'étape 1 appelle récursivement $d \times 145$, avec ATTEINT$=\{a, b, c, h\}$. Les maxmods non dominants sont : $\{c\}$ et $\{e\}$. Puisque c est dans ATTEINT, on engendre seulement le concept $de \times 14$.

Etape 13 : On traite $de \times 14$, avec ATTEINT$=\{a, b, c, h\}$. Le seul maxmod non dominant est : $\{b, c\}$. Puisque b et c sont dans ATTEINT, on ne peut engendrer aucun nouveau concept. La pile de récursivité est vide et l'algorithme prend fin.
◇

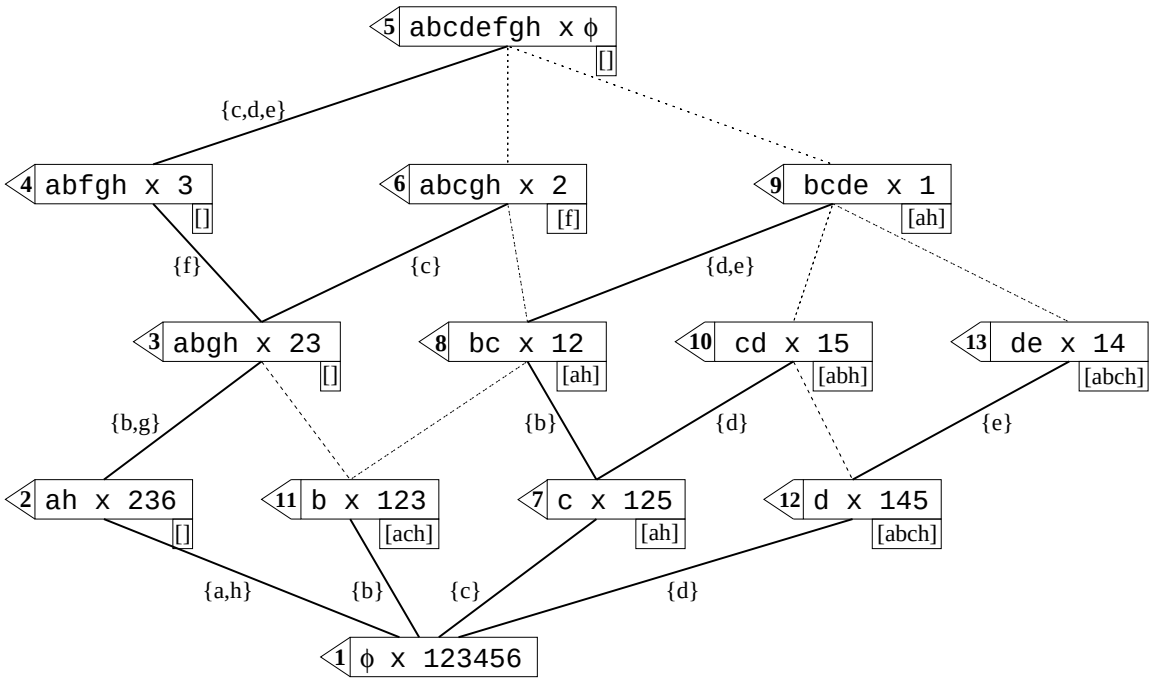


FIG. 6.14 – Le treillis des concepts $\mathcal{L}(\mathcal{R})$ de la relation $\mathcal{R}$ de l'exemple 6.1.5.

Les concepts sont numérotés dans l'ordre préfixe en suivant l'exécution récursive de l'exemple 6.2.6. Les ensembles hérités de sommets déjà traités apparaissent entre crochets. Chaque arête de l'arbre recouvrant défini par l'exécution est étiquetée par le maxmod non dominant utilisé pour calculer le concept suivant.

En conclusion, cette nouvelle approche algorithmique pour la génération de concepts nous permet d'améliorer tous les algorithmes existant pour ce problème. En effet, on utilise un espace mémoire de taille faiblement polynomiale, avec une complexité temporelle comparable à celle de [NR99] et significativement meilleure que celle de [Gan84]. Par ailleurs, dans la mesure où l'on considère que la taille du treillis est fonction directe de la taille de la relation, m est d'ordre n pour les treillis de taille exponentielle, notre complexité devient alors $O(n^2)$ par chaîne maximale parcourue. Enfin, notre analyse de complexité pourrait certainement être affinée, car $O(nm)$ par chaîne maximale dans le cas général reste une estimation très grossière du coût du processus de mise à jour.

Une autre approche prometteuse pour améliorer la complexité de ce problème pourrait être d'utiliser la propriété disant que les chaînes maximales d'un treillis des concepts sont en bijection avec les triangulations du graphe co-biparti sous-jacent (voir [BS02a]) et d'appliquer les travaux récents de Meister (voir [Mei02]) sur les triangulations en temps linéaire des graphes AT-free et Claw-free (une sur-classe des graphes co-bipartis).

Conclusion de la deuxième partie

Notre codage d'une relation binaire par un graphe non orienté, doublée de la mise en évidence de l'ordre de domination, a permis d'aboutir à des résultats, à la fois sur la mise à jour des sous-hiérarchies de Galois, étudiées dans les applications de génie logiciel, et sur la génération efficace des concepts définis par une relation binaire.

Les mêmes outils interviennent dans les travaux en cours, qui concernent l'extraction de règles d'association.

CONCLUSIONS GENERALES ET PERSPECTIVES

Nos travaux coïncident avec un regain général d'intérêt pour les graphes, lié à la démonstration de la conjecture forte des graphes parfaits, à l'introduction si longtemps attendue des graphes dans les programmes de l'enseignement secondaire, à une reconnaissance des graphes comme une branche des mathématiques que Claude Berge revendiquait depuis toujours.

Les graphes offrent des outils puissants qui ont beaucoup évolué depuis dix ans et nous pensons avoir aidé, par notre contribution, à montrer comment on pouvait mettre à profit ces résultats pour concevoir des algorithmes spécifiques à des domaines applicatifs qui sont actuellement d'une importance cruciale.

Nos travaux ont été bien reçus, puisque, pour la première partie, Martin Golumbic nous a proposé la mise en place d'un projet qui permettra de poursuivre cette direction de recherche, et que, pour la partie Analyse Formelle de Concepts, la communauté nous a accueillis en acceptant nos papiers et en nous proposant des problématiques supplémentaires ainsi que des collaborations.

Notre travail de recherche ne fait donc que commencer. Il nous reste de multiples voies à explorer, tant en bio-informatique, par l'analyse expérimentale et la généralisation de nos résultats à des données non phylogénétiques, qu'en analyse formelle de concepts, où les propriétés de l'ordre de domination débouchent sur des problématiques comme l'extraction de la base de règles d'associations et les rapports étroits entre relation binaire et logique propositionnelle.

Bibliographie

[BM70] M. Barbut & B. Monjardet. *Ordre et classification.* Classiques Hachette, 1970.

[Bar02] F. Barnabé. *Décomposition d'une matrice de dissimilarité.* Mémoire de DEA, sous la direction d'Anne Berry, Université Clermont-Fd II, septembre 2002.

[BG91] J–P. Barthélémy, A. Guénoche. *Les arbres et les représentations de proximité.* Masson (Ed.), 1988.

[BLM84] J–P. Barthélémy, B. Leclerc, B. Monjardet. *Ensembles ordonnés et taxonomie.* Annals of Discrete Mathematics, Elsevier Science Publishers B.V. 5north-Holland), 23 :523–548, 1984

[Ber95] A. Berry. *Treillis de Galois des séparateurs minimaux d'un graphe non-orienté.* Mémoire de DEA, Université Montpellier II, 1995.

[Ber98] A. Berry. *Désarticulation d'un graphe.* Thèse de l'Université Montpellier II, 1998.

[Ber99] A. Berry. *A Wide-Range Efficient Algorithm for Minimal Triangulation.* Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99), Baltimore, p. 860–861, Jan. 1999.

[BB97] A. Berry & J–P. Bordat. *Clique minimal separator decomposition.* Rapport de Recherche n°97–213, LIRMM, 1997.

[BB98] A. Berry & J–P. Bordat. *Separability generalizes Dirac's theorem.* Discrete Applied Mathematics, 84 :43–53, 1998.

[BB99] A. Berry & J–P. Bordat. *Orthotreillis et séparabilité dans un graphe non-orienté.* Mathématiques, Informatique et Sciences Humaines, 146 :5–17, 1999.

[BB00] A. Berry & J–P. Bordat. *Triangulated and Weakly Triangulated Graphs : Simpliciality in Vertices and Edges.* Communication, 6th International Conference on Graph Theory (ICGT'00), Marseille, septembre 2000.

[BB01] A. Berry & J–P. Bordat. *Asteroidal triples of moplexes.* Discrete Applied Mathematics, 111 : 219-229, 2001.

[BBC00] A. Berry, J–P. Bordat & O. Cogis. *Generating all the minimal separators of a graph.* International Journal of Foundations of Computer Science, 11 :397-404, 2000.

[BBH00] A. Berry, J–P. Bordat & P. Heggernes. *Recognizing weakly triangulated graphs by edge separablity.* Nordic Journal of Computing, 7 :164–177, 2000.

[BBS02] A. Berry, A. Bordat & A. Sigayret. *Efficient concept generation.* Rapport de Recherche n° 02-105, LIMOS - Université Clermont-Fd II, septembre 2002. En cours de soumission.

[BG00] V. Berry & O. Gascuel. *Inferring evolutionary trees with strong combinatorial evidence.* Theoretical Computer Science, 240(2) :271–298, 2000.

[BS02a] A. Berry & A. Sigayret. *Representing a Concept Lattice by a Graph.* Workshop on Discrete Mathematics for Data Mining (DMDM), Proc. 2nd SIAM Conference on Data Mining (SDM'02), Arlington (VA, USA), 11-13 avril 2002. (submitted to D.A.M.)

[BS02b] A. Berry & A. Sigayret. *Obtaining and maintaining polynomial-sized concept lattices.* Workshop on Formal Concept Analysis for Knowledge Discovery in Data Bases (FCAKDD), Proc. European Conference on Artificial Intelligence (ECAI'02), Lyon (France), juillet 2002.

[BS02c] A. Berry & A. Sigayret. *Maintaining Class Membership Information.* Managing of Specialization/Generalisation Hierarchies (MASPEGHI'02), LNCS Proc. of Conference on Object-Oriented Information System (OOIS'02), Montpellier (France), septembre 2002.

[BSS02d] A. Berry, A. Sigayret & C. Sinoquet. *Maximal sub–triangulation as improving phylogenetic data.* Rapport de Recherche n°02-102, LIMOS - Université Clermont-Fd II, 2002.

[BSS02e] A. Berry, A. Sigayret & C. Sinoquet. *Towards improving phylogeny reconstruction with combinatorial-based constraints on an underlying family of graphs.* Rapport de Recherche n°02-103, LIMOS - Université Clermont-Fd II, 2002.

[BSS02f] A. Berry, A. Sigayret & C. Sinoquet. *Using the threshold family of graphs to improve phylogenetic data.* Poster, Journées Ouvertes Biologie Informatique et Mathématiques (JOBIM'02), Saint–Malo, Juin 2002. (*basé sur le rapport de recherche [BSS02e].*)

[Bir67] G. Birkhoff. *Lattice Theory.* American Mathematical Society, Provicence, 3rd (new) Edition, 1967.

[Bit00] M. Bittner & al.. *Molecular classification of cutaneous malignant meloma by gene expression profiling.* Nature, 406, 536, 3 août 2000.

[BHT01] J.R.S. Blair, P. Heggernes & J.A. Telle. *A practical algorithm for making filled graphs minimal.* Theoretical Computer Science, 250 :124–141, 2001.

[Bor86] J–P. Bordat. *Calcul pratique du treillis de Galois d'une correspondance.* Mathématiques, Informatique et Sciences Humaines, 96 :31–47, 1986.

[Bor92] J–P. Bordat. *Sur l'algorithmique combinatoire d'ordres finis.* Thèse d'état, Université Montpellier II - Sciences et Techniques du Languedoc, 1992

[BLS99] A. Brandstädt, V.B. Le & J.P. Spinrad. *Graph Classes : a Survey.* SIAM Monographs on Discrete Mathematics and Applications, 1999.

[BB87] G. Brassard & P. Bratley. Algorithmique : conception et analyse. Presses de l'Université de Montréal, Masson, 1987.

[Bun71] P. Buneman. *The recovery of trees from measures of dissimilarity.* Mathematics in the Archeological and Historical Sciences, 387–395, Edinburgh University Press, 1971.

[Bun74] P. Buneman. *A characterization of rigid circuit graphs.* Discrete Mathematics, 9 :205–212, 1974.

[Che69] M. Chein. *Algorithme de recherche de sous-matrices premières d'une matrice.* Bull. Math. R.S. Roumanie, 13, 1969.

[CL96] J–B. Chen & S.C. Lee. *Generation and Reorganization of Subtype Hierarchies.* Journal of Object-Oriented Programming, 8(8), 1996.

[COS95] D.G. Corneil, S.Olariu & L.S. Stewart. *Computing a dominating pair in an astroidal triple free graph in linear time.* Lecture Notes in Computer Science, 955 :358–368, 1995.

[COS97] D.G. Corneil, S.Olariu & L.S. Stewart. *Asteroidal Triple Free Graphs.* SIAM Journal of Discrete Mathematics, 10 :399-430, 1997.

[DHMO94] E. Dalhaus, P. Hammer, F. Maffray & S. Olariu. *On Domination Elimination Orderings and Domination Graphs.* Proc. 20th International Workshop on Graphs (WG'94), Mershing (Germany), 16–18 juin 1994 ; LNCS, 903 :81–92, 1994.

[Dir61] G.A. Dirac. *On rigid circuit graphs.* Abh. Math. Sem. Univ. Hamburg, 25 :71–76, 1961.

[Gan84] B. Ganter. *Two basic algorithms in concept analysis.* Preprint 831, Technische Hochschule Darmstadt, juin 1984.

[GW99] B. Ganter & R. Wille. *Formal Concept Analysis (Mathematicals Foundations).* Springer-Verlag Berlin Heidelberg, 1999.

[GG00] H. Garetta, A. Guénoche. *Quelle confiance accorder à une représentation arborée ?* Actes des Journées Ouvertes Biologie Informatique et Mathématiques (JOBIM'00) ; Lecture Notes in Computer Science, O. Gascuel and M.-F. Sagot (Ed.), Springer Verlag, p 45–56, 2001.

[Gav74] F. Gàvril. *The intersection graphs of subtrees of trees are exactly the chordal graphs.* Journal Combinatorial Theory B, 16 :47–56, 1974.

[GM93] R. Godin & H. Mili. *Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices.* Proc. of ACM OOPSLA'93, Special issue of Sigplan Notice, 28(10) :394–410, 1993.

[GMMM95] R. Godin, G. Mineau & R. Missaoui, H ; Mili. *Conceptual Clustering methods based on Galois lattices and applications.* Revue d'Intelligence Artificielle, 9(2), 1995.

[Gol80] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.

[Gue93] A. Guénoche. *Construction du treillis de Galois d'une relation binaire.* Mathématiques, Informatique et Sciences Humaines, 121 :23–34, 1993.

[Gue98] A. Guénoche. *Ordinal Properties of tree distances.* Discrete Mathematics, 192 :103–117, 1998.

[Hay85] R. Hayward. *Weakly triangulated graphs.* J. Comb. Theory, 39 :200–208, 1985.

[Hay96] R. Hayward. *Generating weakly triangulated graphs.* J. Graph Theory, 21 :67–70, 1996.

[Hei89] J. Hein. *An optimal algorithm to reconstruct trees from additive distance data.* Bulletin of mathematical biology, 51(5) :597-603, 1989.

[HM93] P. Hammer & F. Maffray. *Preperfect graphs.* Combinatorica, 13 :199–208, 1993.

[HMS01] D. Hand, H. Mannila & P. Smyth. *Principles of Data Mining.* MIT Press, 2001.

[HHM89] R. Hayward, C. Hoáng, and F. Maffray. *Optimizing weakly triangulated graphs.* Graphs and Combinatorics, 5 :339–349, 1989.

[HSS00] R. Hayward, J. Spinrad & R. Sritharan. *Weakly chordal graph algorithms via handles.* Proc. 11th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'00), 2000.

[HsM99] W-L. Hsu & T-H. Ma. *Substitution decomposition on chordal graphs and its applications.* SIAM Journal on Computing, 28 :1004-1020, 1999.

[Huch02] M. Huchard. Mémoire d'Habilitation à Diriger les Recherches, Université Montpellier II, à paraître : novembre 2002.

[HDL00] M. Huchard, H. Dicky & H. Leblanc. *Galois lattice as a framework to specify building class hierarchies algorithms.* Theoretical Informatics and Applications, 34 :521–548, 2000.

[HNW99] D. Huson, S. Nettles & T. Warnow. *Obtaining highly accurate topology estimates of evolutionary trees from very short sequences.* Proc. RECOMB'99, Lyon (France), 198-207, 1999.

[Iba00] L. Ibarra. *Fully dynamic algorithms for chordal graphs and split graphs.* University of Victoria, Technical report, DCS-262-IR, 2000.

[KHM97] P. Kearney, R. Hayward, H. Meijer. *Inferring evolutionary trees from ordinal data.* Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97), 418-426, 1997.

[KK98] T. Kloks & D. Kratsch. *Listing all minimal separators of a graph.* SIAM Journal of Computing, 27 :605–613, 1998.

[KKS93] T. Kloks, D. Kratsch & J. Spinrad. *Treewidth and pathwidth of cocomparability graphs of bounded dimension.* Research Report 93-46, Eindhoven University of Technology, 1993.

[KO02] S.O. Kuznetsov & S.A. Obiedkov. *Comparing Performance of Algorithms for Generating Concept Lattices.* Journal for Experimental and Theoretical Artificial Intelligence (JETAI), to appear

[Lei93] H–G. Leimer. *Optimal decomposition by clique separators.* Discrete Mathematics, 113 :99–123, 1993.

[LB62] C.G. Lekkerkerker & J.Ch. Boland. *Representation of a finite graph by a set of intervals on Real Line.* Fund. Math., 51 :45–64, 1962.

[LS98] M. Liquière & J. Sallantin. *Structural machine learning with Galois lattices and graphs.* Proc. International Conference on Machine Learning (ICML'98), Morgan Kaufmann Ed.,p 305–313, 1998.

[Mei00] D. Meister. *Minimale Triangulationen AT-freier Graphen.* Diplomarbeit Zur Erlangung des academischen Grades Diplom–Informatiker, Institut für Informatik, Friedrich-Schiller-Universität Jena, Octobre 2000.

[Mei02] D. Meister. *Minimal triangulation for some classes of AT-free graphs.* Satellite-workshop of WG'02, Prague (Rep. Tchèque), juin 2002.

[MN98] E. Mephu Nguifo & P. Njiwoua. *Using Lattice-based Framework as a Tool for Feature Extraction.* ECML, p 304–309, 1998.

[MN93] M. Morvan & L. Nourine. *Sur la distributivité du treillis des antichaînes maximales d'un ensemble ordonné* C.R. Aca. Sc. ParisV, t 319 série I, 1115–1120, 1993

[Nor78] E.M. Norris. *An algorithm for computing the maximal rectangles of a binary relation.* Revue Roumaine de Mathématiques Pures et Appliquées, 23(2) :243–250, 1978.

[NR99] L. Nourine & O. Raynaud. *A Fast Algorithm for building Lattices.* Information Processing Letters, 71 :199–204, 1999.

[OCF76] T. Ohtsuki, L. K. Cheung & T. Fujisawa. *Minimal triangulation of a graph and optimal pivoting order in a sparse matrix.* Journal of Math. Analysis and Applications, 54 :622–633, 1976.

[Ore44] O. Ore. *Galois connections.* Transactions of the A.M.S., 55 :494–513, 1944.

[Par96] A. Parra. *Structural and algorithmic aspects of chordal graph embeddings.* PhD Dissertation, Technische Universität Berlin, 1996.

[PS95] A. Parra & P. Scheffler. *How to use the minimal separators of a graph for its chordal triangulation.* Proc. 22nd International Colloquium on Automata, Languages and Programming (ICALP '95) ; Lecture Notes in Computer Science, 944 :123–134, 1995.

[Pas00] N. Pasquier. *Data Mining : algorithmes d'extraction et de réduction des règles d'associations dans les bases de données.* Thèse de doctorat, Université Clermont-Fd II, 2000.

[RG02] V. Ranwez & O. Gascuel. *TripleML : une amélioration des méthodes de distance pour l'inférence phylogénétique, grâce à une approche locale du maximum de vraisemblance basée sur les triplets.* Actes de JOBIM'02, p 37–46, Journées Ouvertes Biologie Informatique et Mathématiques, Saint-Malo (France), 10-12 juin, 2002.

[Rob69] F.S. Roberts. *Indifference Graphs.* Proof Techniques in Graph Theory, F. Harary (Ed.), Academic Press NY, p 139–146, 1969.

[RTL76] D. Rose, R.E. Tarjan & G. Lueker. *Algorithmic aspects of vertex elimination on graphs.* SIAM Journ. Comput., 5 :146–160, 1976.

[She99] H. Shen. *Separators are as simple as cutsets.* Asian Computer Science Conference, Purket, Thailand, December 10-13, 1999 ; Lecture Notes in Computer Science, 172 :347–358, 1999.

[SL97] H. Sheng & W. Liang. *Efficient enumeration of all minimal separators in a graph.* Theoretical Computer Science, 180 :169–180, 1997.

[SS95] J. Spinrad & R. Sritharan. *Algorithms for weakly triangulated graphs.* Discrete Applied Mathematics, 59 :181–191, 1995.

[Tar85] R.E. Tarjan. *Decomposition by clique separators.* Discrete Mathematics, 55 :221–232, 1985.

[TY84] R. E. Tarjan & M. Yannakakis. *Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs.* SIAM Journal of Computing, 13 :566–579, 1984.

[Tod99] I. Todinca. *Aspects algorithmiques des triangulations minimales des graphes.* Thèse de troisième cycle, Ecole Normale Supérieure Lyon, 1999.

[T-Rex] V. Makarenkov & P. Casgrain. *Logiciel T-Rex (v. 2.0a1/Win.32), Tree and Reticulogram Reconstruction Software.*, Téléchargeable gratuitement sur le site en ligne de Philippe Casgrain au département de sciences biologiques de l'Université de Montréal, URL : "http ://fas.umontreal.ca/biol/casgrain/en/labo/t-rex/", téléchargé en décembre 2000.

[VMG02] P. Valtchev, R. Missaoui & R. Godin. *A Framework for Incremental Generation of Frequent Closed Item Sets.* Workshop on Discrete Applied Mathematics for Data Mining, Proc. 2nd SIAM Workshop on Data Mining (SDM'02), Arlington (VA, USA), 11-13 avril 2002.

[Wal72] J.R. Walter. *Representations of Rigid Circuit Graphs.* Ph.D. Dissertation, Wayne State University, Detroit, 1972.

Document édité sous LaTeX2e, avec MikTeX 1.20d (TexShell et Yap).

Index

2-paire, 64

ADD-SUB-TRI
  algorithme, 68
  Exemple d'exécution, 69
AjouterCroix
  Algorithme, 127
Algorithme
  `ADD-SUB-TRI`, 68
  `AjouterCroix`, 127
  `CONCEPTS-1`, 147
  `CONCEPTS-2`, 151
  `ORDOM`, 139
  `POND`, 146
  `PostTraitement`, 152
  `PréTraitement`, 152
  `RetirerCroix`, 128
  `SHG`, 141
Arête, 34
Arborescence, 19
Arbre, 36
  d'évolution, 53
  phylogénétique, 53
Arc, 18
Ascendant, 21
ATF (graphe), 39
Atome, 25

Borne
  inférieure, 24
  supérieure, 24
Bottom, 24

Chaîne, 19, 23
  maximale, 19, 23
Chemin, 35
Circuit, 19
Claw-free (graphe), 39
Clique, 35
Co-atome, 25
Comparabilité, 22
Complément, 26, 27
Composante
  connexe, 35
  pleine, 41
Composition
  de graphe, 36
Concept (formel), 32
CONCEPTS-1
  Algorithme, 147
  Exemple d'exécution, 148
CONCEPTS-2
  Algorithme, 151
  Exemple d'exécution, 153
Configuration (d'une famille), 71
Connexité, 35
Contexte (formel), 32
Contraction (de sommets), 36
Corde, 35
Correspondance de Galois, 29
Couverture, 21
Croix, 16
Cycle, 35

Décomposition, 43
  d'un co-biparti, 104
  d'un graphe, 36
  d'une phylogénie, 79
  d'une relation, 104
  par séparateurs, 79
Degré (d'un sommet), 34
DESCENDANCE (algorithme), 20
Descendant, 19, 21
deux-paire, 64
Diagramme de Hasse, 22
Dissimilarité, 52
Distance, 52
  additive d'arbre, 52
  arborée, 52
  triangulée, 61
Dominant(e)
  Ensemble, 113
  Paire, 114

Sommet, 114
Domination, 114
dans un co-biparti, 118
Ensemble dominant, 113
faible, 114
forte, 114, 115
large, 115
Ordre, 117
Paire dominante, 114
Reconstituer l'ordre, 139
Relation, 114
Sommet dominé, 114
Sommet dominant, 114
stricte, 115
Dualité
Ordres, 22
Treillis, 27
Treillis de Galois, 30

Ecart, 52
Ensemble ordonné, 21
Equivalence (relation d'), 18
Extension, 29, 33

Famille de graphes
emboîtés, 58
Isomorphisme, 71
Fermé, 29
Fermeture, 28
de Galois, 29
réflexo-transitive, 22

Galois
Correspondance, 29
Fermeture, 29
Graphe, 35
AT-free, 39
biparti
non orienté, 37
orienté, 19
co-biparti, 37
Composition, 36
Composition par arêtes, 64
Contraction, 36
d'intervalles, 39
d'intervalles propres, 39
Décomposition, 43
Elimination, 36
emboîté, 58
faiblement triangulé, 38
Isomorphisme, 71
localement split, 79
non orienté, 34, 35
biparti, 37
orienté, 18
biparti, 19
Parcours, 20
Parcours, 35
sans griffe, 39
sans trou, 38
seuil (emboîté), 58
seuil (threshold), 58
sous-jacent, 88
Sous-triangulation, 67
split, 79
Structure articulatoire, 95
triangulé, 37
Griffe, 39

Hasse (diagramme de), 22
Hole-free (graphes), 38

Inégalité triangulaire, 52
Incidence (matrice d'), 19, 35
Indice
d'écart, 52
de distance, 52
Inf(), 24
Inf-irréductible, 25
Instanciation, 54
Intension, 29, 33
Introducteur, 132
Introduire
un objet, 132
une propriété, 132
Irréductible, 25

LB-TRIANG (algorithme), 46
Lex-BFS (algorithme), 37
LEX-M (algorithme), 46

Maj(), 24
Majorant, 24
Matrice
Graphe non orienté, 35
Graphe orienté, 19
instanciable, 54
ordinale, 55
intègre, 55

Matrice ordinale, 54
Maximal (élément), 24
Maximum, 24
Maxmod, 35
MCS (algorithme), 37
MCSM (algorithme), 46
Min(), 24
Minimal (élément), 24
Minimum, 24
Minorant, 24
Module, 35
  complet, 35
  maximal, 35
  Maxmod, 35
Moplex, 121

Objet, 32
ORDOM
  Algorithme, 139
  Exemple d'exécution, 141
Ordonnance, 54
Ordre, 21
  canonique, 21
  Comparabilité, 22
  gradué, 23
  ipsodual, 22
  large, 21
  quotient, 21
  Sous-ordre, 22
  strict, 21
  total, 23
Orthocomplément, 27
Orthocomplémentation, 27

Parcours
  Graphe non orienté, 35
  Graphe orienté, 20
PARCOURS (algorithme), 20
Pavé, 29
Phylogénie, 53
POND
  Algorithme, 146
  Exemple d'exécution, 147
PostTraitement
  Algorithme, 152
Prédécesseur, 18, 21
Préordonnance, 54
Préordre, 18
Prétraitement
  Algorithme, 152
Propriété, 32

Racine, 19
Rectangle maximal, 29
Relation, 16
  acyclique, 17
  antisymétrique, 17
  binaire, 16
  complémentaire, 17
  complète, 33
  d'équivalence, 18
  duale, 17
  interne, 17
  irréflexive, 17
  partielle, 16
  réciproque, 17
  Réduire, 30
  réduite, 30
  réflexive, 17
  Sous-relation, 16
  symétrique, 17
  Table, 16
    simplifiée, 18
  transitive, 17
  vide, 33
Représentation arborée, 52
RetirerCroix
  Algorithme, 128
RTL (algorithme), 46

Séparateur, 40
  ab-séparateur, 40
  minimal, 40
Séparateurs croisants, 42
Saturation, 45
  d'un ensemble de sommets, 35
Schéma
  d'élimination, 36
  de composition, 36
Schéma inductif, 28
SHG
  Algorithme, 141
Sommet, 18, 34
  isolé, 35
  simplicial, 35
  universel, 35
Sous-hiérarchie
  de Galois, 132

sur les intensions, 134
Sous-ordre, 22
Sous-triangulation maximale, 67
Stable, 35
Structure articulatoire, 95
Successeur, 18, 21
Sup(), 24
Sup-irréductible, 25

T-Rex (logiciel), 54
Table
D, $D_{\mathscr{P}}$, 130
T, $T_{\mathscr{P}}$, 130
d'une relation, 16
de cardinalité, 130
de domination, 126
L_{dom}, 126
simplifiée, 18
Threshold (graphe), 58
Top, 24
Topologie arborée, 53
Transition (entre graphes), 71
Treillis, 25
booléen, 27
complémenté, 26
complet, 26
de Galois, 30
de séparabilité, 32
des concepts, 32
distributif, 26
orthocomplémenté, 27
simplifié, 132
Triangulation, 45
d'un co-biparti, 91
minimale, 45, 102
Triplet astéroïdal, 39
Trou, 38

Un, 16

Voisin, 34
extérieur, 34
Voisinage, 34

Zéro, 16