Décompositions fonctionnelles et structurelles dans les modèles graphiques probabilistes appliquées à la reconstruction d'haplotypes

Aurélie Favier

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse III Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :
Informatique
spécialité Inteliigence Artificelle

Présentée et soutenue par :
Aurélie FAVIER

le : lundi 12 décembre 2011

Titre :

Décompositions fonctionnelles et structurelles dans les modèles graphiques probabilistes appliquées à la reconstruction d'haplotypes

---

Ecole doctorale :
Mathématiques Informatique Télécommunications (MITT)

Unité de recherche :
Biométrie et Intelligence Artificielle

Directeur(s) de Thèse :

Simon de GIVRY et Andrés LEGARRA

Rapporteurs :

Mme Christine SOLNON (INSA, Lyon)
Mme Pascale LEROY (INRA, Rennes)

Membre(s) du jury :

M. Martin COOPER (IRIT, Toulouse), Président
M. Christophe GONZALES (LIP6, Paris)
M. Philippe JEGOU (LSIS, Marseille)
Mme Maria MARTINEZ (INSERM, Toulouse)
M. Jean-Michel ELSEN (INRA, Toulouse), Invité

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse III Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :
Informatique
spécialité Inteliigence Artificelle

Présentée et soutenue par :
Aurélie FAVIER

le : lundi 12 décembre 2011

Titre :

Décompositions fonctionnelles et structurelles dans les modèles graphiques probabilistes appliquées à la reconstruction d'haplotypes

---

Ecole doctorale :
Mathématiques Informatique Télécommunications (MITT)

Unité de recherche :
Biométrie et Intelligence Artificielle

Directeur(s) de Thèse :

Simon de GIVRY et Andrés LEGARRA

Rapporteurs :

Mme Christine SOLNON (INSA, Lyon)
Mme Pascale LEROY (INRA, Rennes)

Membre(s) du jury :

M. Martin COOPER (IRIT, Toulouse), Président
M. Christophe GONZALES (LIP6, Paris)
M. Philippe JEGOU (LSIS, Marseille)
Mme Maria MARTINEZ (INSERM, Toulouse)
M. Jean-Michel ELSEN (INRA, Toulouse), Invité

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse III Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :
Informatique
spécialité Inteliigence Artificelle

Présentée et soutenue par :
Aurélie FAVIER

le : lundi 12 décembre 2011

Titre :

Décompositions fonctionnelles et structurelles dans les modèles graphiques probabilistes appliquées à la reconstruction d'haplotypes

---

Ecole doctorale :
Mathématiques Informatique Télécommunications (MITT)

Unité de recherche :
Biométrie et Intelligence Artificielle

Directeur(s) de Thèse :

Simon de GIVRY et Andrés LEGARRA

Rapporteurs :

Mme Christine SOLNON (INSA, Lyon)
Mme Pascale LEROY (INRA, Rennes)

Membre(s) du jury :

M. Martin COOPER (IRIT, Toulouse), Président
M. Christophe GONZALES (LIP6, Paris)
M. Philippe JEGOU (LSIS, Marseille)
Mme Maria MARTINEZ (INSERM, Toulouse)
M. Jean-Michel ELSEN (INRA, Toulouse), Invité

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse III Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :
Informatique
spécialité Inteliigence Artificelle

Présentée et soutenue par :
Aurélie FAVIER

le : lundi 12 décembre 2011

Titre :

Décompositions fonctionnelles et structurelles dans les modèles graphiques probabilistes appliquées à la reconstruction d'haplotypes

---

Ecole doctorale :
Mathématiques Informatique Télécommunications (MITT)

Unité de recherche :
Biométrie et Intelligence Artificielle

Directeur(s) de Thèse :

Simon de GIVRY et Andrés LEGARRA

Rapporteurs :

Mme Christine SOLNON (INSA, Lyon)
Mme Pascale LEROY (INRA, Rennes)

Membre(s) du jury :

M. Martin COOPER (IRIT, Toulouse), Président
M. Christophe GONZALES (LIP6, Paris)
M. Philippe JEGOU (LSIS, Marseille)
Mme Maria MARTINEZ (INSERM, Toulouse)
M. Jean-Michel ELSEN (INRA, Toulouse), Invité

THESE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITE DE TOULOUSE

délivrée par l'Université de Toulouse III - Paul Sabatier

ECOLE DOCTORALE MATHEMATIQUES INFORMATIQUE TELECOMMUNICATION

DOMAINE DE RECHERCHE : Intelligence Artificielle

Présentée par

Aurélie Favier

Décompositions fonctionnelles et structurelles dans les modèles graphiques probabilistes appliquées à la reconstruction d'haplotypes

Directeurs de thèse : **M. Simon de Givry, M. Andrés Legarra**

Soutenue le 12 décembre 2011

Devant la Commission d'Examen

JURY

M. Martin Cooper	Professeur des Universités	Président
Mme. Pascale Leroy	Directrice de Recherches INRA	Rapporteur
Mme. Christine Solnon	Professeur des Universités	Rapporteur
M. Christophe Gonzales	Professeur des Universités	Examinateur
M. Philippe Jégou	Professeur des Universités	Examinateur
Mme. Maria Martinez	Directrice de Recherches INSERM	Examinateur
M. Jean-Michel Elsen	Directeur de Recherches INRA	Invité

Remerciements

Je tiens tout d'abord à remercier Mesdames Christine Solnon et Pascale Leroy qui ont accepté d'être les rapporteurs de cette thèse. Je remercie également tous les membres du jury Madame Maria Martinez, Messieurs Martin Cooper, Christophe Gonzales, Philippe Jégou et Jean-Michel Elsen.

Je vous remercie grandement, Simon de Givry et Andrés Legarra, pour m'avoir entourée à chaque étape de cette thèse, d'avoir toujours été disponibles malgré des journées bien remplies. Merci d'avoir eu confiance en moi lorsque moi-même je ne l'étais pas ! Ces trois années furent passionnantes et très agréables.

Je voudrais remercier l'ensemble des personnes des unités BIA et SAGA pour leur accueil, en particulier, Pascale, Jackie et Nathalie pour votre efficacité dans toutes les parties obscures de l'administration.

Une colocation, ce n'est jamais simple, chacun essaie de trouver sa place, alors quatre dans 12 m^2, il aurait pu y avoir des dégâts ... Merci à mes colocataires du bureau BIA 20, Ronan, Eric et Mahuna pour les avoir limités et la bonne ambiance que nous avons su conserver ! Les éternelles confrontations philosophiques Eric/Mahuna nous auront quand même bien fait rire ! Merci à Jimmy et Mathieu d'avoir compléter notre petit groupe pour encore plus de rigolade.
Ne t'inquiète pas Carine je pense à toi fortement, la seule personne féminine dans cette histoire de thésards !!! Nos "petites" discussions de début de soirée dans ton bureau vont me manquer, bonne chance à Limoges !

Malgré tout il y a une vie en dehors de l'Inra et j'ai fait d'incroyables rencontres. Ma Juju je suis toujours prête pour être ton cobaye culinaire. Merci à Marion, Lucie, Raphaëlle et Raphaël pour nos apéros-réconforts après nos leçons d'équitation pas toujours excellentes !!! A Prince, Folie, Festina et Balestan pour m'avoir fait oublier la réalité de la thèse à chaque cours.

Malgré tout, avant de démarrer une thèse il faut bien passer un master adéquat, merci Cédric de m'avoir embarquée avec toi. Et finalement c'est un peu grâce à toi que je peux écrire ces lignes.

Mention spéciale à Cyril pour être à mes côtés chaque jour, et m'encourager. Ta gentillesse et ta patience m'ont permise d'être plus positive.

Un grand merci à l'ensemble de ma famille qui de près ou de loin m'a toujours soutenue. Une petite pensée pour mon grand-père Jacques qui aurait certainement apprécié ce résultat.

A ma sœurette Constance qui m'a manquée malgré nos appels réguliers, à mon frère Sylvain qui, pourquoi pas, écrira une thèse à son tour.

Finalement, je ne serai pas là où j'en suis sans mes parents. Vous avez toujours su comment me faire avancer, c'est avec quelques larmes que je vous embrasse fort, merci.

A Cyril,
A mes parents.

Table des matières

Introduction

L'amélioration des races d'animaux d'élevage se fait actuellement par l'analyse de l'information génétique des individus dans le but de sélectionner ceux qui présentent des caractères économiques importants comme la quantité de la production laitière, la fertilité ou encore la performance sportive.

Pour réaliser cette sélection, il faut localiser sur le génome l'effet quantitatif choisi. A l'heure actuelle, il existe un processus biologique expérimental permettant de lire l'information génétique, le *génotype*, à des positions précises le long des chromosomes. Mais ces informations ne sont pas suffisamment précises, il est nécessaire de connaître l'origine parentale de chaque partie de cette information génétique : une partie du génotype provient du père des individus et l'autre de la mère de l'individu.

De manière générale, il y a deux situations dans lesquelles la reconstruction des haplotypes est étudiée : soit à partir de génotypages d'individus supposés non-apparentés, soit à partir d'individus qui forment une structure de famille(s). Dans les deux cas, il s'agit, soit de maximiser un critère de parcimonie, soit de maximiser la vraisemblance du jeu de données.

Dans le cas d'une population avec des relations de famille (pedigree), il est possible d'établir avec plus ou moins de certitude selon les cas quelle est l'origine (maternelle ou paternelle) et la coségrégation des marqueurs génétiques. Ceci est fondé sur un principe d'indépendance conditionnelle simple, la méïose chez le père est indépendante de la méïose chez la mère.

Le problème de reconstruction d'haplotypes est alors formulé en terme de modèles probabilistes graphiques comme les réseaux bayésiens [Friedman et al., 2000, Lauritzen and Sheehan, 2003]. Or la communauté Intelligence Artificielle a développé de puissantes techniques pour la résolution des réseaux bayésiens et dans un cadre proche que sont les réseaux de fonctions de coûts. Un réseau bayésien peut être transformé en un réseau de fonctions de coûts dont les techniques de résolutions sont méconnues dans la communauté de génétique statistique.

Pour exemples, les méthodes classiques de reconstruction d'haplotypes appliquent généralement une séquence d'élimination (*peeling* en génétique) de groupe d'individus [Elston and Stewart, 1971] ou de marqueurs génétiques [Lander and Green, 1987] qui s'apparente fortement à l'élimination de variables [Dechter, 1999] en programmation dynamique (utilisée dans les réseaux bayésiens et les réseaux de contraintes). De même, l'algorithme d'élimination de génotypes [Lange and Goradia, 1987]

peut être assimilé à la cohérence d'arc généralisée [Mackworth, 1977] dans les réseaux de contraintes.

Le but de cette thèse va être d'utiliser et d'enrichir ces techniques autour des réseaux de fonctions de coûts pour faire de la reconstruction d'haplotypes dans les pedigrees.

Lorsqu'une tâche est trop complexe à réaliser, il est naturel de vouloir la décomposer en plusieurs petites tâches simples. Pour les réseaux de fonctions de coûts, il existe plusieurs niveaux de décomposition : dont entre autres la décomposition de la structure et la décomposition des fonctions de coûts.

Les cohérences locales ont démontré au cours de ces dernières années leur intérêt pour l'optimisation dans les réseaux de fonction de coûts. Mais elles ne peuvent être utilisées uniquement sur des fonctions de coûts de petites arités. Nous avons élaboré une décomposition des fonctions de coûts permettant de les exprimer sous forme de sommes de fonctions d'arité inférieure [Favier et al., 2011b]. Cette décomposition s'est avérée efficace pour réduire la formulation de la reconstruction d'haplotypes dans le cas des pedigrees de demi-frères et nous a conduit à identifier et formaliser une modélisation compacte de ce problème [Favier et al., 2010].

Avant d'aborder l'inférence dans les réseaux de fonctions de coûts nous nous sommes intéressés à un problème de comptage de solutions, ce dernier étant un problème très difficile dans le cadre d'un calcul exact, nous avons utilisé une décomposition du réseau en plusieurs sous-réseaux suivant des propriétés structurelles. Nous avons élaboré une méthode de comptage exact, pouvant être utilisée efficacement sur les sous-problèmes décomposés pour obtenir une approximation du nombre de solutions du réseau de départ [Favier et al., 2011a].

Structure du manuscrit

La première partie est une introduction des formalismes de modèles graphiques probabilistes auxquels nous nous intéressons ainsi que ceux utilisés dans la littérature pour traiter la reconstruction d'haplotypes (les chaînes de Markov en particulier). Le chapitre 1 détaille le vocabulaire de la théorie des graphes. Le chapitre 2 présente deux modèles probabilistes que sont les réseaux bayésiens et les chaînes de Markov, leur formalisme, les requêtes qu'ils peuvent traiter ainsi que quelques algorithmes d'inférence et d'optimisation. Le dernier chapitre de cette partie est consacré aux réseaux de fonctions de coûts à leur définition et à leurs techniques de optimisation.

La deuxième partie est constituée de deux chapitres détaillant les contributions de cette thèse pour le traitement des réseaux de fonctions de coûts consacrée à leur décomposition pour l'inférence ou l'optimisation. Le chapitre 4 présente une décomposition de fonctions de coûts permettant de les exprimer sous forme de somme de fonctions contenant moins de variables. Nous montrons que cette décomposition est équivalente à l'indépendance par paire dans les modèles graphiques probabilistes. Le chapitre 5 présente l'utilisation d'une décomposition structurelle pour traiter le problème de comptage de solutions.

La troisième et dernière partie est consacrée à la reconstruction d'haplotype fil conducteur de cette thèse. Le chapitre 6 introduit les notions de génétique nécessaires à la bonne compréhension du problème. Le chapitre 7 formalise le problème de reconstruction d'haplotypes et présente quelques méthodes parmi les nombreuses existantes. Le chapitre 8 décrit notre formulation du problème pour les pedigrees particuliers de demi-frères, pouvant être obtenus de manière automatique grâce à la décomposition fonctionnelle décrite au chapitre 4 et aux technqiques existantes des réseaux de fonctions de coûts.

Partie I

Etat de l'art : Modèles graphiques probabilistes

Notions de la théorie des graphes 1

Ce chapitre a pour but de présenter les différentes notions relatives aux graphes, qui seront largement utilisées dans les chapitres qui suivent. Les graphes sont utilisés pour représenter les différents modèles dits graphiques (les réseaux bayésiens, les réseaux de contraintes par exemple). Généralement les sommets de ces graphes seront associés aux variables des modèles et les liens reliant les sommets seront associés aux différentes fonctions (tables de probabilités conditionnelles ou de transitions, fonctions de coûts, contraintes ...) présentes dans les modèles. La théorie développée sur les graphes est importante, ici nous n'en présentons qu'une petite partie, elle est utilisée dans certains algorithmes des modèles graphiques pour les simplifier, les décomposer, établir des propriétés structurelles sur ces modèles.

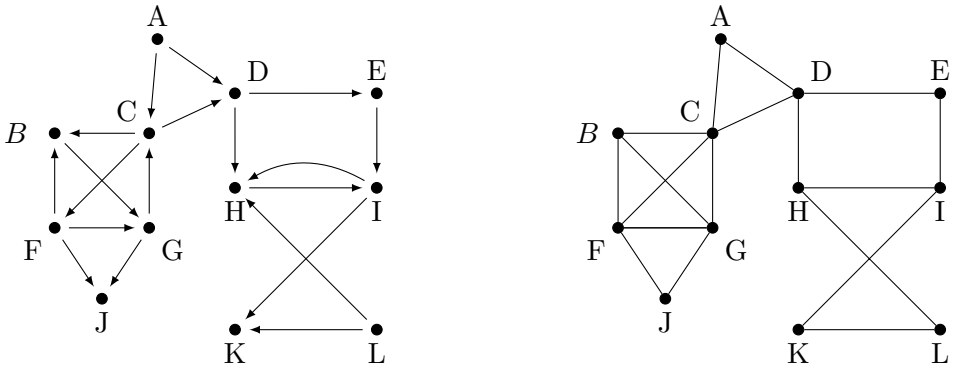
1.1 Définitions générales

Définition 1.1 Graphe
Un *graphe* $G = (\mathbf{S}, \mathbf{A})$ est une paire de deux ensembles, un ensemble de sommets $\mathbf{S} = \{s_1, \ldots, s_n\}$. et un ensemble d'arêtes $\mathbf{A} = \{a_1, \ldots, a_m\}$. Une arête est un ensemble de deux sommets $\{s_i, s_j\}$, $1 \leqslant i \neq j \leqslant n$.

Définition 1.2 Graphe orienté
Un graphe orienté $G = (\mathbf{S}, \mathbf{A})$ est une paire de deux ensembles, un ensemble de *sommets* $\mathbf{S} = \{s_1, \ldots, s_n\}$. et un ensemble d'*arcs* $\mathbf{A} = \{a_1, \ldots, a_m\}$. Un arc est un couple de deux sommets $(s_i \to s_j)$.

Les figures 1.1a et 1.1b représentent respectivement un graphe orienté et non orienté dont l'ensemble des sommets est $\{A, B, \ldots, K, L\}$ pour les deux graphes.

Définition 1.3 Sous-graphe
Soient un graphe (orienté ou non) $G = (\mathbf{S}, \mathbf{A})$ et $\mathbf{X}$ un ensemble de sommets tel que $\mathbf{X} \subset \mathbf{S}$. Le



(a) Graphe orienté **(b)** Un graphe non orienté.

Figure 1.1 – Graphes

sous-graphe de G induit par $\mathbf{X}$ est le graphe $G[\mathbf{X}] = (\mathbf{X}, \mathbf{A_X})$ où $\mathbf{A_X}$ est défini par :

$\mathbf{A_X} = \{(s_i \to s_j) \in \mathbf{A} \mid s_i, s_j \in \mathbf{X}\}$ (Graphe orienté)

$\mathbf{A_X} = \{\{s_i, s_j\} \in \mathbf{A} \mid s_i, s_j \in \mathbf{X}\}$ (Graphe non orienté)

Définition 1.4 Graphe partiel

Soient un graphe (orienté ou non) $G = (\mathbf{S}, \mathbf{A})$ et $\mathbf{E}$ un ensemble d'arêtes (ou d'arcs) tel que $\mathbf{E} \subset \mathbf{A}$. Le *graphe partiel de G induit par* $\mathbf{E}$ est le graphe $G[\mathbf{E}] = (\mathbf{S}, \mathbf{E})$.

Définition 1.5 Graphe complet

Un graphe $G = (\mathbf{S}, \mathbf{A})$ est dit *complet* si et seulement si pour toute paire de sommets s_i et s_j, l'arête $\{s_i, s_j\}$ appartient à $\mathbf{A}$.

Définition 1.6 Clique

Soit $G = (\mathbf{S}, \mathbf{A})$ un graphe. Une *clique* est un ensemble de sommets $\mathbf{X} \subseteq \mathbf{S}$ tel que tous les sommets sont reliés entre eux c'est-à-dire que le sous-graphe $G[X]$ est un graphe complet.

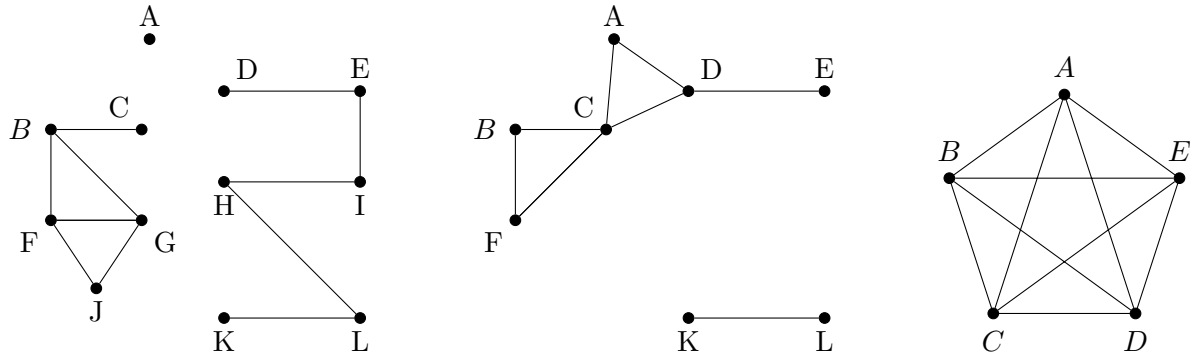
La taille de la clique est le nombre de sommets qui la composent. Une clique est dite *maximale* si elle n'est pas strictement incluse dans une autre clique. Dans le graphe la figure 1.1b, l'ensemble $\{B, C, F, G\}$ forme une clique maximale du graphe.

La figure 1.2a représente le graphe partiel de G induit par l'ensemble des arêtes {B,C}, {B,F}, {B,G}, {F,G}, {F,J}, {G,J}, {D,E}, {E,I}, {H,I}, {H,L} et {K,L}. La figure 1.2b représente le sous-graphe de G induit par l'ensemble des sommets A,B,C,D,E,F,K et L.

Le graphe G de la figure 1.2c est un graphe complet à cinq sommets A, B,C, D et E.

1.2 Parcours, Voisinage et Connexité

Par la suite, nous nous intéressons à un graphe orienté.



(a) Un graphe partiel du graphe 1.1b **(b)** Un sous-graphe du graphe 1.1b **(c)** Le graphe complet à cinq sommets G

Figure 1.2 – Graphes non orientés

1.2.1 Parcours

Définition 1.7 Chemin

Un *chemin* entre deux sommets s_i et s_j dans G est une séquence d'arcs de la forme $(a_{u_1}, \ldots, a_{u_{k+1}})$ où $a_{u_l} = (s_{u_{l-1}} \to s_{u_l})$

$\forall l,\ 1 \leqslant l \leqslant k+1,\ a_{ul} \in \mathbf{A},\ a_{u_1} = (s_i \to s_{u_1})$ et $a_{u_{k+1}} = (s_{u_k} \to s_j)$. Ce chemin sera noté $[s_i s_{u_1} s_{u_2} \ldots s_{u_k} s_j]$ en donnant la séquence des sommets associés aux arcs composant le chemin.

Un chemin est élémentaire s'il ne passe pas deux fois par le même sommet.

La figure 1.1a représente un graphe orienté à douze sommets $\{A, \ldots, L\}$. La séquence [B G C D E] forme un chemin du sommet B au sommet E avec les arcs $(B \to G),\ (G \to C),\ (C \to D)$ et$(D \to E)$.

Définition 1.8 Circuit

Un *circuit* est un chemin dont le sommet de départ est également le sommet d'arrivée.

Définition 1.9 Chaîne

Un *chaîne* entre deux sommets s_i et s_j dans G est une séquence d'arcs de la forme $(a_{u_1}, \ldots, a_{u_{k+1}})$ où $a_{u_l} = (s_{u_{l-1}} \to s_{u_l})$ ou $a_{ul} = (s_{u_l} \to s_{u_{l-1}})$

$\forall l,\ 1 \leqslant l \leqslant k+1,\ a_{u_l} \in \mathbf{A},\ a_{u_1} = (s_i \to s_{u_1})$ ou $a_{u_1} = (s_{u_1} \to s_i)$ et $a_{u_{k+1}} = (s_{u_k} \to s_j)$ ou $a_{u_{k+1}} = (s_j \to s_{u_k})$. Cette chaîne sera notée $[s_i s_{u_1} s_{u_2} \ldots s_{u_k} s_j]$ en donnant la séquence des sommets associés aux arcs la composant.

Une chaîne est élémentaire si elle ne passe pas deux fois par le même sommet. Tout chemin est une chaîne, mais toute chaîne n'est pas un chemin.

[J G C D] est une chaîne du graphe 1.1a composée des arcs $(G \to J)$, $(G \to C)$ et $(C \to D)$. Toute fois elle n'est pas un chemin du graphe puisqu'il n'existe pas d'arc $(J \to G)$.

Définition 1.10 Cycle
Un *cycle* est une chaîne de longueur non nulle dont le sommet d'origine est le même que le sommet d'arrivée et pour lequel un même arc n'est pas utilisé deux fois.

Tout circuit est un cycle mais la réciproque est fausse.

Définition 1.11 Longueur et distance
La *longueur* d'un chemin (chaîne) est égale au nombre d'arcs qui le (la) compose. La *distance* entre deux sommets s_i et s_j est égale à la longueur du chemin le plus court entre s_i et s_j.

1.2.2 Voisinage

Définition 1.12 Parents et ascendance
Soit un graphe orienté $G = (\mathbf{S}, \mathbf{A})$ tel que $s_i, s_j \in \mathbf{S}$ et $(s_i \to s_j) \in \mathbf{A}$. s_i est un *parent* de s_j.
L'ensemble des parents de s_j est défini par $\texttt{parents}(s_j) = \{s \in \mathbf{S} \mid (s \to s_j)\}$.
L'ascendance de s_j dans G est définie par $Asc(s_j) = \{s \in \mathbf{S} \mid \exists \text{ chemin de } s \text{ vers } s_j\}$

Définition 1.13 Enfants et descendance
Soit un graphe orienté $G = (\mathbf{S}, \mathbf{A})$ tel que $s_i, s_j \in \mathbf{S}$ et $(s_i \to s_j) \in \mathbf{A}$. s_j est un *enfant* de s_i.
L'ensemble des enfants de s_i est défini par $\texttt{enfants}(s_i) = \{s \in \mathbf{S} \mid (s_i \to s)\}$.
La descendance de s_i dans G est définie par $Desc(s_i) = \{s \in \mathbf{S} \mid \exists \text{ chemin de } s_i \text{ vers } s\}$

Définition 1.14 Voisin et voisinage
Soit un graphe orienté $G = (\mathbf{S}, \mathbf{A})$. s_i et s_j deux sommets du graphe sont voisins si et seulement si $(s_i \to s_j)$ ou $(s_j \to s_i)$ sont des arcs du graphe. Le voisinage de s_i est l'ensemble de ses voisins définis par : $\texttt{voisins}(s_i) = \texttt{parents}(s_i) \cup \texttt{enfants}(s_i)$

Définition 1.15 Degré
Soit un sommet s du graphe G.
Le degré sortant du sommet s est le nombre des enfants de s, $\delta^+(s) =\mid \texttt{enfants}(s) \mid$.
Le degré entrant du sommet s est le nombre des parents de s, $\delta^-(s) =\mid \texttt{parents}(s) \mid$.
Le degré du sommet s est défini par $\delta(s) = \delta^+(s) + \delta^-(s)$.

Pour le graphe orienté de la figure 1.1a, nous avons $\texttt{parents}(I) = \{E, H\}$, $\texttt{enfants}(I) = \{H, K\}$ et $\delta(I) = 4$.

Remarque 1.16 *Pour les graphes non orientés les notions de voisin, chaîne et cycle sont définies en remplaçant "arc" par "arête". Mais les notions relatives à l'orientation des arcs comme les parents, les enfants ou les degrés sortants et entrants perdent leur sens dans le cas des graphes non orientés.*

1.2.3 Connexité

Définition 1.17 Composante connexe
Une composante connexe dans un graphe quelconque, est un ensemble maximal de sommets tel que pour toute paire de sommets s_i et s_j, appartenant à cet ensemble il existe une chaîne reliant s_i à s_j.

Un graphe est dit *connexe* si tous ses sommets sont dans la même composante connexe.

Le graphe de la figure 1.2a n'est pas connexe. Une de ses composantes connexes se compose des sommets $\{D, E, H, I, K, L\}$.

1.3 Quelques classes de graphes

Plusieurs classes de graphes ont été définies par rapport à leurs propriétés structurelles. Les trois classes présentées sont celles que nous retrouverons plus tard, dans les chapitres suivants.

1.3.1 Graphes sans circuit

Les graphes sans circuit sont utilisés par exemple pour représenter les graphes associés aux réseaux bayésiens, aux chaînes de Markov, ces notions seront d'ailleurs présentées dans le chapitre 2.

Définition 1.18 Soit un graphe $G = (\mathbf{S}, \mathbf{A})$, une *source* de G est un sommet s de $\mathbf{S}$ tel que $\texttt{parents}(s) = \emptyset$. Un *puits* de G est un sommet p de $\mathbf{S}$ tel que $\texttt{enfants}(p) = \emptyset$.

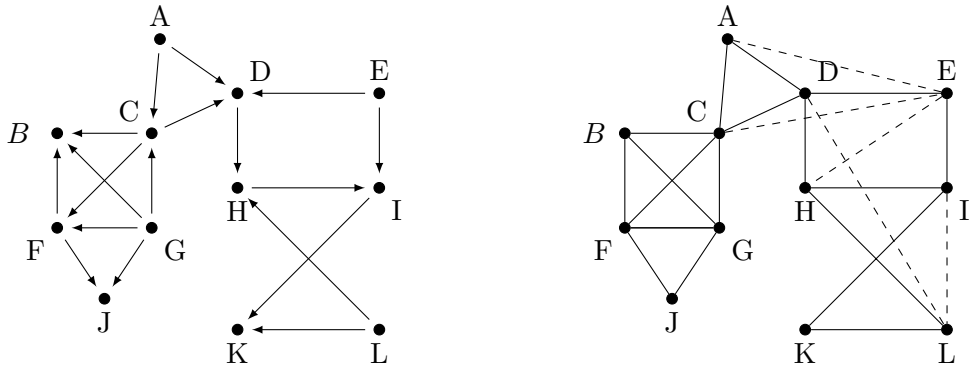
Propriété 1.19 *Tout graphe sans circuit admet au moins une source et un puits.*

Propriété 1.20 *Tout sous-graphe d'un graphe sans circuit est sans circuit.*

A partir d'un graphe orienté sans circuit, il est possible de construire un graphe non orienté, en supprimant l'orientation des arcs et en reliant ("mariant") les parents de chaque sommet entre eux, ce graphe est appelé *graphe moral*.

Définition 1.21 Graphe moral
Soit $G = (\mathbf{S}, \mathbf{A})$ un graphe sans circuit. Le graphe moral $G^m = (\mathbf{S}, \mathbf{A}^m)$ de G est non orienté tel que
- Si $(s_i \rightarrow s_j) \in \mathbf{A}$ alors $\{s_i, s_j\} \in \mathbf{A}^m$
- Pour chaque sommet s, si $s_i, s_j \in \texttt{parents}(s)$ dans G alors $\{s_i, s_j\} \in \mathbf{A}^m$.

La figure 1.3 représente un graphe sans circuit (1.3a). Le sommet D possède trois parents : A,C,E, dans le graphe moral(1.3b) associé ces trois sommets sont reliés par des arêtes. Il en va de même pour les parents D, L du sommet H et I,L du sommet K et H,E du sommet de I.



(a) Un graphe sans circuit. **(b)** Son graphe moral.

Figure 1.3 – Un graphe sans circuit et son graphe moral.

1.3.2 Graphes acycliques

Définition 1.22 Arbre
Un *arbre* est un graphe $T = (\mathbf{S}, \mathbf{A})$ connexe et sans cycle.

La notion d'arborescence est l'adaptation de la définition d'un arbre pour les graphes orientés connexes et acycliques.

Définition 1.23 Polyarbre
Un polyarbre est un graphe orienté $T = (\mathbf{S}, \mathbf{A})$, connexe et acyclique tel qu'il existe au maximum une chaîne entre deux sommets de $\mathbf{S}$.

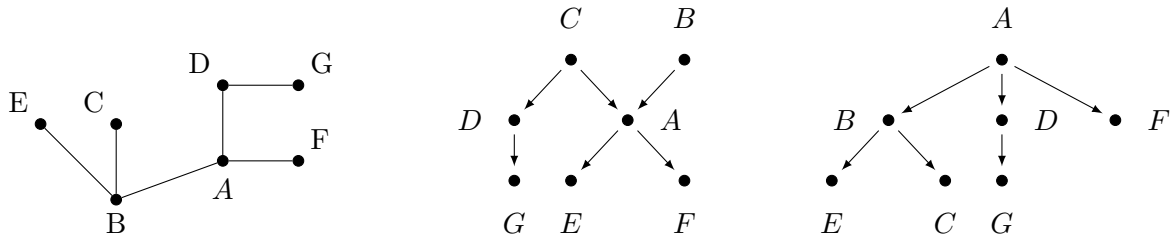
Définition 1.24 Arborescence
Une *arborescence* est un polyarbre tel que chaque sommet s n'a qu'un parent qu'on notera $\texttt{Pere}(s)$ et un seul sommet r appelé *racine* admet comme descendance $\mathbf{S}$, $Desc(r) = \mathbf{S}$.

Les sommets n'ayant pas de descendance sont généralement appelés les *feuilles* de l'arbre ou de l'arborescence. La figure 1.4 illustre ces différents graphes acycliques définis.

Définition 1.25 Sous-arborescence
Soient $T = (\mathbf{S}, \mathbf{A})$ est une arborescence et $s \in \mathbf{S}$, le sous-graphe $T[s \cup Desc(s)]$ est appelé sous-arborescence de T enracinée en s.

1.3.3 Graphes triangulés

Définition 1.26 Graphe triangulé
Un graphe non orienté $G = (\mathbf{S}, \mathbf{A})$ est *triangulé* si et seulement si chaque cycle de longueur supérieure



(a) Un arbre **(b)** Un polyarbre **(c)** Une arborescence

Figure 1.4 – Graphes acycliques

ou égale à quatre possède une corde c'est-à-dire deux sommets non consécutifs dans le cycle sont voisins.

Propriété 1.27 *Tout sous-graphe d'un graphe triangulé est triangulé.*

Le graphe de la figure 1.1b page 8 n'est pas triangulé car dans le cycle [D E I H D] les sommets D et I ne sont pas voisins dans le graphe ni les sommets E et H. De même, le cycle [H I K L H] n'admet pas de corde. Par contre dans celui de la figure 1.5a pour ces mêmes cycles, il existe bien des cordes, ici $\{E, H\}$ et $\{I, L\}$, ainsi que pour tous les autres cycles, le graphe est triangulé.

Définition 1.28 Soit un graphe $G = (\mathbf{S}, \mathbf{A})$, un sommet s de G est *simpliciel* si et seulement si le sous-graphe $G[\texttt{voisins}(s)]$ est complet.

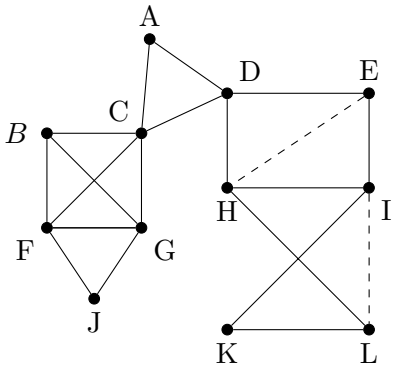
Définition 1.29 Soit $\mathcal{O} = [s_1, \ldots, s_n]$ un ordre sur les sommets du graphe $G = (\mathbf{S}, \mathbf{A})$. On appelle voisinage ultérieur d'un sommet s_i l'ensemble $\texttt{voisins}^+(s_i) = \{s_j \in \texttt{voisins}(s_i) \mid j > i\}$

Définition 1.30 Un ordre $\mathcal{O} = [s_1, \ldots, s_n]$ sur les sommets du graphe $G = (\mathbf{S}, \mathbf{A})$ est un schéma parfait d'élimination si et seulement si chaque s_i est simpliciel pour le sous-graphe $G[\texttt{voisins}^+(s_i)]$ c'est-à-dire $G[\texttt{voisins}^+(s_i)]$ est un graphe complet.

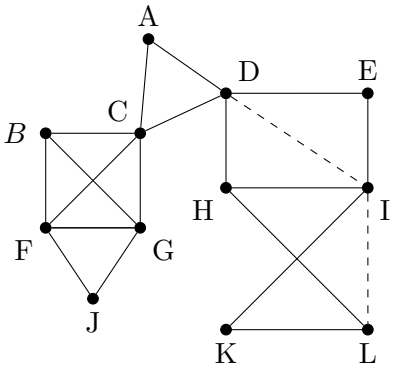
Il en découle une équivalence entre les graphes triangulés et les schémas parfaits d'élimination.

Théorème 1.31 ***[Fulkerson and Gross, 1965]***
Un graphe $G = (\mathbf{S}, \mathbf{A})$ est triangulé si et seulement si G possède un schéma parfait d'élimination.

On utilise le procédé de *triangulation* pour obtenir un graphe triangulé à partir d'un graphe quelconque $G = (\mathbf{S}, \mathbf{A})$. Un ensemble d'arêtes $\mathbf{N}$ est ajouté tel que $G' = (\mathbf{S}, \mathbf{A} \cup \mathbf{N})$ soit un graphe triangulé. En pratique, nous construisons un ordre sur les sommets du graphe à trianguler. La triangulation revient alors à ajouter les arêtes nécessaires pour que l'ordre choisi soit un schéma parfait d'élimination du graphe.



(a) Graphe triangulé avec l'ordre Min-fill.



(b) Graphe triangulé avec l'ordre MCS.

Figure 1.5 – Résultat de la triangulation du graphe 1.1b page 8 par les heuristiques Min-Fill et MCS

Il existe des heuristiques pour construire l'ordre des sommets à partir duquel le graphe va être triangulé. Voici deux exemples d'heuristiques très utilisées pour la triangulation :

MCS (Maximum Cardinality Search) [Tarjan and Yannakakis, 1984] Elle ordonne les sommets de n à 1, en choisissant arbitrairement le premier sommet, le sommet suivant étant pris parmi ceux ayant le plus grand nombre de voisins déjà numérotés. Sa complexité est en $O(n + m')$ où m' est le nombre d'arêtes dans G'.

Min-Fill Elle ordonne les sommets de 1 à n, en sélectionnant comme nouveau sommet, celui qui permettra d'ajouter un minimum d'arêtes si l'on complète le sous-graphe induit par ses voisins non encore numérotés. Sa complexité est en $O(n(n + m'))$.

Suivant l'ordre Min-Fill [B A J F G C D E H I K L] la triangulation du graphe de la figure 1.1b page 8 donnée par la figure 1.5a . Suivant l'ordre MCS [K L E I H J B G F A D C] le résultat de la triangulation est représentée par le graphe de la figure 1.5b .

1.4 Décomposition arborescente

Définition 1.32 Décomposition arborescente

Etant donné $G = (\mathbf{S}, \mathbf{A})$ un graphe non orienté, une décomposition arborescente de G est un couple $(\mathbf{C}, T)$ où $T = (\mathbf{I}, \mathbf{F})$ est un arbre avec $\mathbf{I}$ l'ensemble des k sommets et $\mathbf{F}$ l'ensemble des arêtes et $\mathbf{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ une famille de sous-ensembles de $\mathbf{S}$ qui vérifie :

- $\bigcup_{i \in \mathbf{I}} \mathcal{C}_i = \mathbf{S}$
- Pour chaque arête $\{x, y\} \in \mathbf{A}$, il existe $i \in \mathbf{I}$ avec $\{x, y\} \subseteq \mathcal{C}_i$.
- Pour tout $i, j, k \in \mathbf{I}$, si k est sur une chaîne allant de i à j dans T, alors $\mathcal{C}_i \cap \mathcal{C}_j \subset \mathcal{C}_k$.

Les C_i sont appelés *clusters*.

Cette définition ne donne pas une décomposition unique. Nous pouvons évaluer son degré de ressemblance à un arbre par la largeur associée, ainsi plus celle-ci est proche de 1 (largeur d'un arbre) plus la ressemblance est importante.

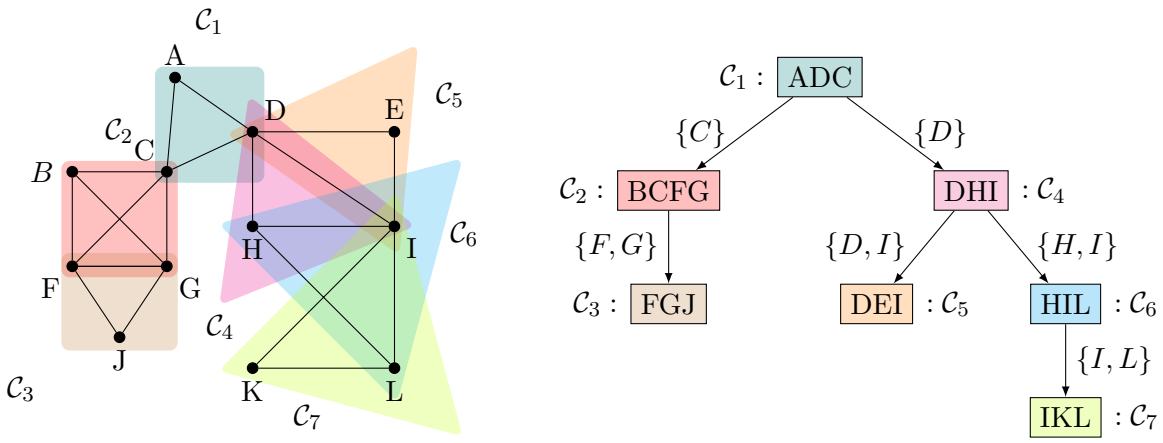
Définition 1.33 Largeur d'arborescence
La *largeur* d'une décomposition arborescente $(\mathbf{C}, T)$ de G est définie par $w = \max_{\mathcal{C}_i \in \mathbf{C}} \mid C_i \mid -1$. La *largeur d'arborescence* (ou largeur d'arbre) w^* de G est la largeur minimale de toutes les décompositions arborescentes de G.

Définition 1.34 Séparateurs et sommets propres
Soit une décomposition arborescence $(\mathbf{C}, T)$, on appelle *séparateur* de $\mathcal{C}_i$ le sous-ensemble de variables $S_i = \mathcal{C}_i \cap \mathcal{C}_p$ avec $p = \texttt{Pere}(i)$. L'ensemble des *sommets propres* d'un cluster est noté $V_i = \mathcal{C}_i \setminus S_i$.

Définition 1.35 Descendance d'un cluster
Soit une décomposition arborescente $(\mathbf{C}, T)$. Un cluster $\mathcal{C}_j$ est un *fils* du cluster $\mathcal{C}_i$ si j est un enfant de i dans T. L'ensemble des fils de $\mathcal{C}_i$ est défini par $\texttt{fils}(\mathcal{C}_i) = \{\mathcal{C}_j \mid j \in \texttt{enfants}(i)\}$.

Pour les décompositions arborescentes, la définition de la descendance d'un cluster est un peu différente de celle des graphes. La *descendance* de $\mathcal{C}_i$ est l'ensemble de ses sommets et de ceux des clusters $\mathcal{C}_j$ si j appartient à la descendance de i dans T $\mathcal{C}_i$, $Desc(\mathcal{C}_i) = \bigcup_{j \in Desc(i) \cup i} \mathcal{C}_j$.

Considérons la décomposition arborescente de la figure 1.6b, les fils du cluster $\mathcal{C}_4$ sont les clusters $\mathcal{C}_5$ et $\mathcal{C}_6$ et $Desc(\mathcal{C}_4) = \mathcal{C}_4 \cup \mathcal{C}_5 \cup \mathcal{C}_6 \cup \mathcal{C}_7 = \{D, E, I, H, K, L\}$. La largeur induite de la décomposition est égale à 3. L'ensemble des sommets propres du cluster $\mathcal{C}_2$ est $V_2 = \{B, F, G\}$ et le séparateur avec son père $\mathcal{C}_1$ est $S_2 = \{C\}$.



(a) Recouvrement en clusters

(b) Décomposition arborescente enracinée en $\mathcal{C}_1$ et les séparateurs.

Figure 1.6 – Exemple de décomposition arborescente d'un graphe de la figure 1.5b.

Obtenir une décomposition arborescente optimale, de largeur induite w^*, est un problème NP-difficile. Pour la classe des graphes triangulés, la décomposition est facile à calculer.

Chaque clique maximale d'un graphe triangulé correspond à un cluster de sa décomposition arborescente. Si deux clusters ont des sommets en commun il existe une unique chaîne les reliant.

Ainsi pour calculer une décomposition arborescente d'un graphe, une méthode consiste à le trianguler puis en donner une décomposition.

Exemple 1.36 *Pour obtenir une décomposition arborescente du graphe de la figure 1.1b page 8 nous le triangulons suivant l'heuristique MCS donnée précédemment dont le graphe résultant est à la figure 1.5b page 14. A partir de ce graphe triangulé nous identifions les cliques maximales, mises en évidence par la figure 1.6a, nous avons sept cliques maximales, elles seront les sept clusters de la décomposition arborescente. En choisissant $\mathcal{C}_1$ comme racine de notre arborescence nous obtenons la décomposition arborescente proposée à la figure 1.6b.*

Modèles graphiques probabilistes 2

Ce chapitre a pour but de présenter, de manière superficielle, deux modèles graphiques probabilistes que sont les réseaux bayésiens et les chaînes de Markov. Avant de pouvoir les présenter nous rappelons les notions essentielles de probabilités pour leur compréhension.

2.1 Notions de probabilités

La théorie des probabilités a pour objectif de modéliser ce qui n'est pas déterminé à l'avance (par exemple un lancer de dé). La théorie des probabilités ne va pas permettre de prédire quelle issue va se réaliser mais quelle chance a chaque issue de se réaliser. Ainsi, nous allons associer à chaque issue possible un nombre entre 0 et 1 qui traduit notre estimation des chances que cette issue a de se réaliser : on appelle ce nombre la probabilité de cette issue. On appelle *événement* un ensemble d'issues.

2.1.1 Définitions principales

Définition 2.1 Probabilité
Soient Ω un ensemble fini non vide, $\mathcal{E}$ une famille d'événements observables sur Ω. On appelle *probabilité* sur $(\Omega, \mathcal{E})$ toute fonction $\mathbb{P} : \mathcal{E} \to [0,1]$ vérifiant :

1. $\mathbb{P}(\Omega) = 1$
2. Pour toute suite $(A_i)_{i\geqslant 1}$ d'événements de $\mathcal{E}$ deux à deux disjoints ($\forall i, j\ A_i \cap A_j = \emptyset$) :
   $\mathbb{P}(\bigcup_{i\in\mathbb{N}^*} A_i) = \sum_{i=1}^{+\infty} \mathbb{P}(A_i)$

Le triplet $(\Omega, \mathcal{E}, \mathbb{P})$ s'appelle *espace probabilisé*.

Remarque 2.2 *Ω étant un ensemble fini, se donner une probabilité $\mathbb{P}$ sur l'ensemble des parties de Ω n'est rien d'autre que se donner la fonction $\mathbb{P}$ sur Ω : $\omega \in \Omega \rightarrow p(\omega) = \mathbb{P}(\{\omega\})$. En effet pour tout événement $\mathcal{A}$, on peut écrire $\mathbb{P}(\mathcal{A}) = \sum_{\omega \in \mathcal{A}} p(\omega)$. Cette fonction vérifie les propriétés suivantes :*

- $p : \Omega \rightarrow [0, 1]$
- $\sum_{\omega \in \omega} p(\omega) = 1$

Une telle fonction discrète est une distribution de probabilité.

Propriété 2.3 ***Propriétés générales d'une probabilité***
Toute probabilité $\mathbb{P}$ sur $(\Omega, \mathcal{E})$ vérifie les propriétés suivantes :

1. $\mathbb{P}(\emptyset) = 0$
2. *Additivité*
   a) *Si $A \cap B = \emptyset$, $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$*
   b) *Si les A_i $(1 \leqslant i \leqslant n)$ sont deux à deux disjoints : $\mathbb{P}(\bigcup_i A_i) = \sum_{i=1}^{n} \mathbb{P}(A_i)$*

Définition 2.4 Variable aléatoire
Une *variable aléatoire* est une fonction X définie sur Ω :

$$\begin{array}{rccc} X : & \Omega & \rightarrow & D_X \\ & \omega & \longmapsto & X(\omega) \end{array}$$

Pour $x \in D_X$ on note alors $\{X = x\}$ l'événement $\{\omega \in \Omega \mid X(\omega) = x\}$. D_X est le domaine de définition de X.

Une variable aléatoire n'est rien d'autre qu'une fonction classique. Elle permet de caractériser des événements par une valeur. On parle de variable aléatoire à valeurs discrètes ou variable aléatoire discrète lorsque le domaine de définition de la variable aléatoire est fini.

Dans ce chapitre et dans l'ensemble de cette thèse l'ensemble Ω est fini, les variables aléatoires sont toujours considérées comme discrètes.

Notation 2.5 Pour la suite les variables aléatoires seront représentées par une majuscule (A,B,...). Les valeurs qu'elles prennent lorsque l'issue ω se réalise sont notées par la même lettre en minuscule [1] $(a \in D_A, b \in D_B, \ldots)$.

La plupart des ensembles seront représentés par une lettre en gras $(\mathbf{S}, \mathbf{X}, \ldots)$. Pour les ensembles de variables, nous noterons $D_{\mathbf{X}}$ le produit cartésien des domaines des variables composant l'ensemble $\mathbf{X}$.

1. Lorsqu'aucune ambiguïté n'est possible et pour simplifier au maximum les notations $\mathbb{P}(X(\omega) = x) = \mathbb{P}(\{X = x\}) = \mathbb{P}(X = x) = \mathbb{P}(x)$.

Remarque 2.6 *Il ne faut cependant pas confondre $\mathbb{P}(\mathcal{A})$ la probabilité associée à l'événement $\mathcal{A} \subset \Omega$ et $\mathbb{P}(A)$ qui est la fonction qui associe à tout élément $a \in D_A$ la valeur de probabilité de l'événement $A = a$.*

Définition 2.7 La distribution de la loi de X est $x \longmapsto \mathbb{P}(X = x)$. Pour connaître la loi d'une variable aléatoire il faut connaître l'ensemble de ses valeurs possibles et la probabilité avec laquelle elle réalise chaque valeur.

2.1.2 Probabilités sur plusieurs variables

Définition 2.8 Probabilité jointe
Soient X et Y deux variables aléatoires sur le même univers Ω, la probabilité jointe de X et Y est la fonction définie sur $D_X \times D_Y$ par :

$$\begin{array}{rccc} \mathbb{P}: & D_X \times D_Y & \to & [0,1] \\ & (x,y) & \longmapsto & \mathbb{P}(x,y) \end{array}$$

où $\mathbb{P}(x,y) = \mathbb{P}(\{X = x\} \cup \{Y = y\})$

Cette définition peut être étendue à tout ensemble fini de variables aléatoires.

Définition 2.9 Probabilité marginale
Soient $\mathbf{U}$ un ensemble fini, non vide de variables aléatoires, $\mathbf{V} \subset \mathbf{U}$ un sous-ensemble non vide, $\mathbf{V'} = \mathbf{U} \setminus \mathbf{V}$ et $\mathbb{P}(\mathbf{U})$ la probabilité jointe sur les variables de $\mathbf{U}$. On appelle *marginalisation* de $\mathbb{P}$ sur $\mathbf{V}$ la fonction :

$$\forall \mathbf{v} \in D_{\mathbf{V}},\ \mathbb{P}(\mathbf{v}) = \sum_{\mathbf{v'} \in D_{\mathbf{V'}}} \mathbb{P}(\mathbf{v}, \mathbf{v'})$$

Cette fonction est la probabilité jointe des variables $\mathbf{V}$.

Exemple 2.10 *Soient deux variables aléatoires A et B dont la probabilité jointe est donnée par le tableau suivant :*

$\mathbb{P}(A,B)$	b_1	b_2	b_3
a_1	0.0579	0.1603	0.5118
a_2	0.0771	0.0607	0.1322

Par marginalisation $\mathbb{P}(A = a_1) = \mathbb{P}(A = a_1, B = b_1) + \mathbb{P}(A = a_1, B = b_2) + \mathbb{P}(A = a_1, B = b_3) = 0.73$ d'où les deux probabilités marginales :

	a_1	a_2
$\mathbb{P}(A)$	0.73	0.27

	b_1	b_2	b_3
$\mathbb{P}(B)$	0.136	0.218	0.646

2.1.3 Probabilités conditionnelles, Indépendance

Définition 2.11 Loi fondamentale
Soient deux variables aléatoires sur le même univers. Pour tout $x \in D_X$ et $y \in D_Y$, la probabilité conditionnelle de $X = x$ étant donné $Y = y$ est notée $\mathbb{P}(x \mid y)$ et vérifie :

$$\mathbb{P}(x, y) = \mathbb{P}(x \mid y) \cdot \mathbb{P}(y)$$

La définition suivante généralise la loi fondamentale pour un ensemble de variables aléatoires.

Définition 2.12 Loi fondamentale généralisée
Soit un ensemble de variables aléatoires $(X_i)_{i\in\{1,\dots,n\}}$ sur le même univers.

$$P(x_1, \dots, x_n) = \prod_{i=1}^{n} \mathbb{P}(x_i \mid x_1, \dots, x_{i-1})$$

La convention veut que $\mathbb{P}(X \mid \emptyset) = \mathbb{P}(X)$. Cette définition permet d'énoncer le théorème de Bayes, à la base de la statistique bayésienne et des réseaux bayésiens.

Théorème 2.13 ***Théorème de Bayes***
Soient deux variables aléatoires X et Y. Si $\mathbb{P}(x)$ est non nulle alors

$$\mathbb{P}(y \mid x) = \frac{\mathbb{P}(x \mid y)\mathbb{P}(y)}{\mathbb{P}(x)}$$

La notion de probabilité conditionnelle n'a de sens que si l'événement $Y = y$ influe sur l'événement $X = x$.

Définition 2.14 Indépendance marginale
Deux variables aléatoires X et Y sont *indépendantes* (noté $X \perp\!\!\!\perp Y$) si et seulement si

$$\forall x \in D_X,\ y \in D_Y\ \mathbb{P}(x, y) = \mathbb{P}(x) \cdot \mathbb{P}(y)$$

Cependant deux variables aléatoires peuvent être indépendantes conditionnellement à d'autres, c'est ce qu'on appelle simplement l'*indépendance conditionnelle*.

Définition 2.15 Indépendance conditionnelle
Soient trois variables aléatoires X, Y et Z. X est indépendant de Y conditionnellement à Z (noté $X \perp\!\!\!\perp Y \mid Z$) si et seulement si :

$$\forall x \in D_X,\ y \in D_Y,\ z \in D_Z\ \mathbb{P}(x \mid y, z) = \mathbb{P}(x \mid z)$$

L'indépendance conditionnelle ne dépend pas de l'ordre ainsi nous avons également $\mathbb{P}(y \mid x, z) = \mathbb{P}(y \mid z)$.

L'indépendance marginale n'est autre qu'une indépendance conditionnelle où Z est l'ensemble vide. Plusieurs propriétés découlent de ces indépendances.

Propriété 2.16 *Les propriétés suivantes sont équivalentes*[2] *:*

1. $X \perp\!\!\!\perp Y \mid Z$
2. $\exists F$ telle que $\mathbb{P}(X \mid Y, Z) = F(X, Z)$
3. $\exists G$ telle que $\mathbb{P}(Y \mid X, Z) = G(Y, Z)$
4. $\exists F, G$ telle que $\mathbb{P}(X, Y \mid Z) = F(X, Z) \cdot G(Y, Z)$
5. $\mathbb{P}(X \mid Y, Z) = \mathbb{P}(X \mid Z)$
6. $\mathbb{P}(X, Y \mid Z) = \mathbb{P}(X \mid Z) \cdot \mathbb{P}(Y \mid Z)$
7. $\mathbb{P}(X, Y, Z) = \mathbb{P}(X \mid Z) \cdot \mathbb{P}(Y \mid Z) \cdot \mathbb{P}(Z)$

Théorème 2.17 *Soit un ensemble fini de variables aléatoires* $\{X_1, \ldots, X_n\}$.

$$\mathbb{P}(X_1, \ldots, X_{i-1}) = \prod_{i=1}^{n} \mathbb{P}(X_i \mid \mathbf{V}_i)$$

avec $\forall i,\ \mathbf{V}_i \subset \{X_1, \ldots, X_n\}$ *tel que* $X_i \perp\!\!\!\perp (\{X_1, \ldots, X_{i-1}\} \setminus \mathbf{V}_i) \mid \mathbf{V}_i$

Démonstration.

$$\begin{aligned} P(X_1, \ldots, X_n) &= \prod_{i=1}^{n} \mathbb{P}(X_i \mid X_1, \ldots, X_{i-1}) && \text{loi fondamentale} \\ &= \prod_{i=1}^{n} \mathbb{P}(X_i \mid \mathbf{V}_i, \{X_1, \ldots, X_{i-1}\} \setminus \mathbf{V}_i) \\ &= \prod_{i=1}^{n} \mathbb{P}(X_i \mid \mathbf{V}_i) && \text{propriété 2.16 (5)} \end{aligned}$$

□

2.1.4 Modèles graphiques probabilistes

Les *modèles graphiques probabilistes* (PGM, *Probabilistic Graphical Models*) permettent une approche générale pour les raisonnements automatiques sous incertitude [Koller and Friedman, 2009].

Un *modèle graphique probabiliste* (ou *distribution de Gibbs*) est défini par un produit de fonctions positives $\mathbf{F}$, sur un ensemble de variables discrètes $\mathbf{X}$, exprimant une information probabiliste ou déterministe.

Définition 2.18 Modèle Graphique Probabiliste
Un modèle graphique probabiliste est un triplet $(\mathbf{X}, \mathbf{D}, \mathbf{F})$ avec $\mathbf{X} = \{X_1, \ldots, X_n\}$, un ensemble de variables, $\mathbf{D} = \{D_{X_1}, \ldots, D_{X_n}\}$, un ensemble de domaines finis, et $\mathbf{F} = \{f_1, \ldots, f_m\}$, un ensemble de

2. Dans la suite du manuscrit, on généralise la notation $\forall x, y,\ \mathbb{P}(x, y)$ par $\mathbb{P}(X, Y)$

fonctions à valeurs réelles positives, chacune est définie sur un sous-ensemble de variables $\mathbf{S}_i \subseteq \mathbf{X}$ (i.e. la portée). La distribution jointe est définie par :

$$\mathbb{P}(X_1, \ldots, X_n) = \frac{\prod_{i=1}^{m} f_i(\mathbf{S}_i)}{\sum_{\mathbf{x} \in D_{\mathbf{X}}} \prod_{i=1}^{m} f_i(\mathbf{x}[\mathbf{S}_i])}$$

Les PGM couvrent les réseaux Bayésiens, les réseaux et chaînes de Markov, que nous présentons en partie dans les sections suivantes (2.2 et 2.3) mais également les formalismes déterministes comme les réseaux de contraintes valuées ou non que nous présenterons plus loin dans le chapitre 3.

2.2 Réseaux bayésiens

Un réseau bayésien est une représentation concise de la distribution de probabilité jointe sur un ensemble de variables aléatoires.

Les réseaux bayésiens représentent un cas particulier des PGM utilisant une probabilité conditionnelle par variable ($m = n$) et la constante de normalisation (dénominateur de $\mathbb{P}(\mathbf{X})$) vaut 1.

2.2.1 Définitions

Définition 2.19 Réseau bayésien
Un réseau bayésien est défini par un graphe orienté sans circuit $G = (\mathbf{X}, \mathbf{A})$, un espace probabilisé fini $(\Omega, \mathcal{E}, \mathbb{P})$ et un ensemble de variables aléatoires associées aux nœuds du graphe [3] définies sur $(\Omega, \mathcal{E}, \mathbb{P})$ tels que

$$\mathbb{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbb{P}(X_i \mid \texttt{parents}(X_i))$$

où $\texttt{parents}(X_i)$ l'ensemble des sommets parents de X_i dans G.

Remarque 2.20 *L'ensemble $\mathbf{V}_i$ défini dans le théorème 2.17 page précédente pour un sommet X_i est exactement l'ensemble $\texttt{parents}(X_i)$ dans le cadre des réseaux bayésiens en supposant un ordre topologique sur les variables.*

Les réseaux bayésiens sont généralement définis à partir de variables aléatoires (qui forment les nœuds du graphe) et un ensemble de tables de probabilités conditionnelles.

La figure 2.1 représente un réseau bayésien contenant cinq variables. Il décrit par les saisons de l'année (X_1) si la pluie tombe (X_2), si l'arrosage est en marche (X_3), si le chemin est mouillé (X_4) et si le chemin est glissant (X_5). Toutes ces variables sont binaires. L'absence d'arc allant de X_1 à X_4

3. L'ensemble $\mathbf{X}$ représentera également ces variables aléatoires.

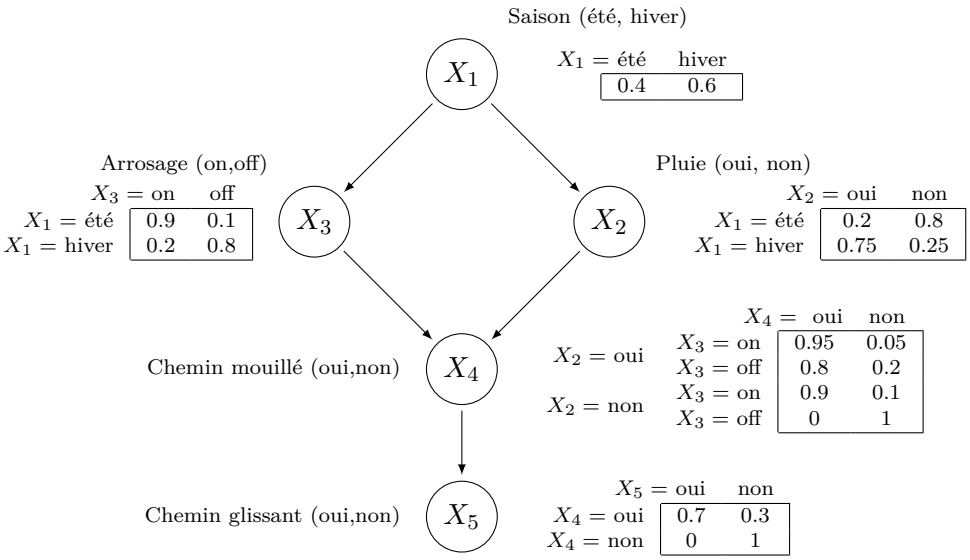


Figure 2.1 – Un réseau bayésien représentant les dépendances entre 5 variables.

signifie que la saison n'influe pas directement sur l'état mouillé ou non du chemin. Ainsi le graphe de la figure 2.1 indique que la probabilité jointe des cinq variables se décompose de la façon suivante :

$$\mathbb{P}(X_1, \ldots, X_5) = \mathbb{P}(X_1) \cdot \mathbb{P}(X_2 \mid X_1) \cdot \mathbb{P}(X_3 \mid X_1) \cdot \mathbb{P}(X_4 \mid X_2, X_3) \cdot \mathbb{P}(X_5 \mid X_4)$$

Les tables de probabilités conditionnelles associées sont données à la figure 2.1. Par exemple, sachant que nous sommes en été (X_1 = été), nous avons une probabilité de 0.2 qu'il pleuve (X_2 = oui).

A partir du graphe associé au réseau bayésien, il est possible de "lire" les indépendances conditionnelles grâce aux propriétés de Markov suivantes.

Nous définissons pour un graphe $G = (\mathbf{X}, \mathbf{A})$ l'ensemble des non-descendants d'un sommet X de G : $nDesc(X) = \{Y \in \mathbf{X} \mid Y \notin Desc(X)\}$ où $Desc(X)$ est l'ensemble des descendants de X.

Propriété 2.21 *Soit un réseau bayésien avec un graphe orienté $G = (\mathbf{X}, \mathbf{A})$ et une loi $\mathbb{P}$. Les propriétés suivantes sont vérifiées.*

Propriété orientée de Markov par paire *La propriété est vérifiée lorsque pour tout X et Y non voisins dans G et $Y \in nDesc(X)$*

$$X \perp\!\!\!\perp Y \mid nDesc(X) \setminus \{Y\}$$

Propriété orientée de Markov locale *La propriété est vérifiée lorsque pour tout $X \in \mathbf{X}$,*

$$X \perp\!\!\!\perp nDesc(X) \mid \texttt{parents}(X)$$

Cependant toutes les indépendances ne peuvent pas toujours être représentées à travers un graphe orienté, dont les indépendances par paire.

Exemple 2.22 *Soit quatre variables aléatoires W, X, Y et Z. Nous avons les indépendances et les non indépendances suivantes :*

$X \perp\!\!\!\perp W \mid Y, Z$

$Y \perp\!\!\!\perp Z \mid X, W$

$X \not\perp\!\!\!\perp Y$

$X \not\perp\!\!\!\perp Z$

$Y \not\perp\!\!\!\perp W$

$Z \not\perp\!\!\!\perp W$.

Il n'est pas possible de représenter toutes ces (in)dépendances avec un graphe orienté, seul un réseau de Markov le peut avec un graphe non orienté.

Soit un graphe non orienté $G = (\mathbf{X}, \mathbf{A})$ et une loi $\mathbb{P}$.

Propriété 2.23 ***Propriété non orienté de Markov par paire***
La propriété est vérifiée lorsque pour chaque X et Y non voisins dans G

$$X \perp\!\!\!\perp Y \mid \mathbf{X} \setminus \{X, Y\}$$

Cette propriété est au cœur du chapitre 4.

Propriété 2.24 ***Propriété non orienté de Markov locale***
La propriété est vérifiée lorsque pour tout $X \in \mathbf{X}$

$$X \perp\!\!\!\perp \mathbf{X} \setminus \overline{\texttt{voisins}(X)} \mid \texttt{voisins(X)}$$

où $\overline{\texttt{voisins}}(X) = \texttt{voisins}(X) \cup \{X\}$

2.2.2 L'inférence

L'inférence consiste à calculer la probabilité *a posteriori* d'une (ou plusieurs) variable(s) du réseau conditionnellement aux variables observées : $\mathbb{P}(X_i \mid \mathbf{O} = \mathbf{o})$ où $\mathbf{O} \subset \mathbf{X}$ est l'ensemble des variables observées. L'inférence a été prouvée comme étant NP-difficile dans le cas général [Cooper, 1990]. Plusieurs algorithmes permettent en théorie de calculer cette probabilité de manière exacte.

2.2.2.1 Algorithmes d'inférence

Propagation de messages [Pearl, 1988] Cette première méthode de propagation de messages n'est valable que pour les réseaux bayésiens ayant comme graphe une arborescence. Cette particularité permet de déduire qu'il existe une unique chaîne entre deux sommets de ce graphe.

Soit $\mathbf{O}$ l'ensemble des variables observées. A partir d'un sommet X, nous pouvons définir deux ensembles de sommets P_X et E_X tels que P_X regroupe les sommets de $\mathbf{O}$ dont la chaîne vers X passe

par un parent de X et E_X regroupe les sommets de $\mathbf{O}$ dont la chaîne vers X passe par un enfant de X et $\mathbf{O} = P_X \cup E_X$. Il est possible de montrer que

$$\mathbb{P}(X \mid \mathbf{O} = \mathbf{o}) \propto \lambda(X) \cdot \pi(X) \tag{2.1}$$

avec $\lambda(X) \propto \mathbb{P}(E_X \mid X)$ et $\pi(X) \propto \mathbb{P}(X \mid P_X)$.

Il s'agit ensuite de calculer des messages λ et π pour chaque variable selon un ordre topologique :

- Les messages λ

Pour chaque enfant Y de X :

$$\lambda_Y(X = x) = \sum_{y \in D_Y} \mathbb{P}(Y = y \mid X = x) \cdot \lambda(Y = y) \tag{2.2}$$

Le calcul de λ pour la variable X s'obtient à l'aide la fonction suivante :

$$\lambda(X = x) = \begin{cases} \delta^x_{x_{\mathbf{O}}} & \text{si } X \text{ est instanciée à } x_{\mathbf{O}} \ (X \in \mathbf{O}) \\ 1 & \text{si } X \text{ est une feuille} \\ \prod\limits_{Y \in \texttt{enfants}(X)} \lambda_Y(X = x) & \text{sinon.} \end{cases} \tag{2.3}$$

où δ^j_i est le symbole de Kronecker [4].

- Les messages π

Pour l'unique parent Z de X :

$$\pi_X(Z = z) = \pi(Z = z) \prod_{U \in \texttt{enfants}(Z) \setminus \{X\}} \lambda_U(Z = z) \tag{2.4}$$

$$\pi(X = x) = \begin{cases} \delta^x_{x_{\mathbf{O}}} & \text{si } X \text{ est instanciée à } x_{\mathbf{O}} \ (X \in \mathbf{O}) \\ \mathbb{P}(X) & \text{si } X \text{ est une racine} \\ \sum\limits_{\mathbf{z} \in D_Z} \mathbb{P}(X = x \mid Z = z) \cdot \pi_X(Z = z) & \text{sinon.} \end{cases} \tag{2.5}$$

dans ce cas sa complexité en temps est $O(nd^2)$ et en espace $O(nd)$ avec $n = \mid \mathbf{X} \mid$, $d = \max_{X_i \in \mathbf{X}} D_{X_i}$

Cette propagation peut également s'effectuer dans les polyarbres [5].

Clique-tree propagation [Lauritzen and Spiegelhalter, 1988, Jensen et al., 1990] La méthode est applicable pour toute structure de graphe sans circuit contrairement à la méthode de propagation de messages précédente. L'idée est de fusionner les variables pour se ramener à un arbre. Elle est divisée en quatre étapes qui sont :

4. Le symbole δ^j_i a la signification suivante : $\begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$

5. graphe acyclique dont les sommets peuvent avoir plusieurs parents.

1. Moralisation du graphe (section 1.3.1 page 11)
2. Triangulation du graphe moral (section 1.3.3 page 12)
3. Construction d'une décomposition arborescente (section 1.4 page 14)
4. Inférence dans la décomposition arborescente en utilisant la propagation de messages précédente

Les clusters de la décomposition arborescente deviennent les variables. L'envoi des messages (les potentiels) dans la décomposition arborescente, permet d'effectuer l'inférence dans n'importe quel réseau bayésien. Les potentiels vont être mis à jour, il sera ensuite possible d'en déduire les lois marginales pour les variables du réseau d'origine.

Cette méthode a une complexité, en temps et en mémoire, exponentielle en la largeur d'arbre.

D'autres méthodes exactes existent, loop cutset conditionning [Pearl, 1988], ainsi que des méthodes approchées basées sur l'échantillonnage avec les méthodes MCMC (pour Markov Chain Monte Carlo) [Gilks et al., 1996] ou l'échantillonnage de Gibbs [Lauritzen, 1996] dont la convergence peut être lente notamment s'il y a des probabilités très faibles. En effet, toutes les probabilités, pour cette dernière méthode, doivent être strictement positives.

2.2.2.2 Requêtes possibles sur la loi de $\mathbb{P}(\mathbf{X})$

Il existe différents types de requêtes pouvant être demandées à partir d'un réseau bayésien de la loi $\mathbb{P}(\mathbf{X})$:

La vraisemblance des observations (PR) Calculer la probabilité $\mathbb{P}(\mathbf{O} = \mathbf{o})$ ayant observé les variables $\mathbf{O} \subset \mathbf{X}$.

Les probabilités marginales (MAR) Calculer la probabilité *a posteriori* $\mathbb{P}(X_i \mid \mathbf{O} = \mathbf{o})$ de toutes les variables $X_i \in \mathbf{X} \setminus \mathbf{O}$.

Maximum a posteriori hypothesis (MAP) Rechercher une affectation partielle $\mathbf{u}$ d'un sous-ensemble $\mathbf{U} \subseteq \mathbf{X} \setminus \mathbf{O}$ de probabilité $\mathbb{P}(\mathbf{U} = \mathbf{u} \mid \mathbf{O} = \mathbf{o})$ maximum.

Most Probable Explanation (MPE) Rechercher une affectation complète $\mathbf{u}$ des variables $\mathbf{U} = \mathbf{X} \setminus \mathbf{O}$ de probabilité $\mathbb{P}(\mathbf{U} = \mathbf{u} \mid \mathbf{O} = \mathbf{o})$ maximum.

Les requêtes PR et MAR sont des problèmes d'inférence, MPE est un problème d'optimisation discrète, la dernière requête MAP est un problème plus difficile qui combine l'optimisation et une forme de comptage. La requête MPE peut également être vue comme un problème d'inférence. La technique est d'effectuer l'inférence en remplaçant les opérations de *sommes* par des opérations *max* dans la propagation de messages [Lauritzen and Spiegelhalter, 1988].

2.3 Chaînes de Markov cachées

Plusieurs méthodes de reconstruction d'haplotypes (*cf.* partie III) font appel à une modélisation sous forme de chaînes de Markov cachées et utilisent leurs algorithmes. Les chaînes de Markov (ca-

chées ou non) sont des cas particuliers de réseaux bayésiens lorsque l'on veut représenter l'évolution temporelle ou spatiale d'une variable aléatoire. La causalité est alors définie d'une variable à l'instant t vers une variable à l'instant $t+1$. Nous nous intéressons ici à des chaînes finies.

2.3.1 Définitions

Définition 2.25 Chaîne de Markov finie
Une chaîne de Markov d'horizon fini,n, est définie par un ensemble de variables aléatoires $\mathbf{X}$ et un espace probabilisé fini $(\Omega, \mathcal{E}, \mathbb{P})$ tel que :

$$\mathbb{P}(X_1, \ldots, X_n) = \mathbb{P}(X_1) \cdot \prod_{t=1}^{n-1} \mathbb{P}(X_{t+1} \mid X_t)$$

Dans le cas des chaînes de Markov on parle également de probabilité de transition de l'état $X_t = e_t$ à l'état $X_{t+1} = e_{t+1}$ à la place de probabilités conditionnelles.

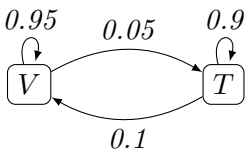
Définition 2.26 Chaîne de Markov cachée
Une chaîne de Markov cachée ou HMM (Hidden Markov Model) $\mathcal{H}$ est définie par

- un ensemble $\mathbf{E}$ de d états dits *cachés*
- un alphabet $\mathcal{D}$ de caractères observés
- une matrice de transition $\mathcal{T} : \mathbf{E} \times \mathbf{E} \to [0,1]$ indiquant la probabilité de passer d'un état e_i à un état e_j notée $\mathbb{P}(e_j \mid e_i)$.
- une matrice de génération $\mathcal{G} : \mathbf{E} \times \mathcal{D} \to [0,1]$ indiquant les probabilités de générer un symbole $o \in \mathcal{D}$ à partir de l'état $e \in \mathbf{E}$ notées $\mathbb{P}(o \mid e)$.
- un vecteur d'initialisation $\Pi : \mathbf{E} \to [0,1]$ indiquant les probabilités initiales $\mathbb{P}(e)$.

Exemple 2.27 *Un casino malhonnête*
Le croupier utilise un vrai dé (V) la plupart du temps, mais parfois utilise un dé truqué (T). Le dé truqué a une probabilité 0.5 pour un 6 et de 0.1 pour chacun des nombres 1 à 5. Le croupier bascule entre un vrai dé et le truqué avec une probabilité de 0.05 et fait le changement inverse avec une probabilité de 0.1. Le HMM correspondant est le suivant :

- $\mathbf{E} = \{V, T\}$
- $\mathcal{D} = \{1, 2, 3, 4, 5, 6\}$
- $\mathcal{T} = \begin{pmatrix} 0.95 & 0.05 \\ 0.1 & 0.9 \end{pmatrix}$
- $\mathcal{G} = \begin{pmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{2} \end{pmatrix}$

Le graphe associé à la probabilité de transition est



A partir d'un HMM les mêmes requêtes que celles des réseaux bayésiens peuvent être réalisées.

Le réseau bayésien associé au HMM est représenté par la figure 2.2, quatre lancés de dé sont représentés.

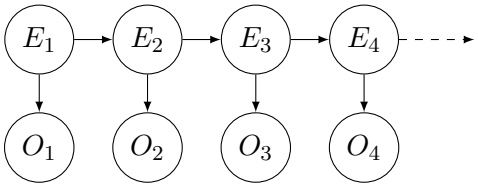


Figure 2.2 – Le graphe du réseau bayésien associé aux HMM

2.3.2 Algorithmes d'inférence

Dans la suite de cette section nous considérons un HMM $\mathcal{H}$.

Si l'on s'intéresse au problème PR, le but est de calculer la probabilité de générer la séquence de symboles $\mathbf{O} = o_1 \dots o_n$ avec le HMM $\mathcal{H}$, l'approche directe est de calculer

$$\mathbb{P}(\mathbf{O}) = \sum_{\{e_1,\dots,e_n\}} \mathbb{P}(e_1) \cdot P(o_1 \mid e_1) \dots \mathbb{P}(e_n \mid e_{n-1}) \cdot \mathbb{P}(o_n \mid e_n)$$

avec $\{e_1, \dots, e_n\}$ un chemin permettant de générer $\mathbf{O}$. Cependant ce calcul s'effectue avec une complexité en $O(nd^n)$. Cette approche n'est clairement pas acceptable, c'est pourquoi des alternatives ont été mises en place à l'instar de l'inférence dans les réseaux bayésiens.

2.3.2.1 Algorithme Forward-Backward

L'algorithme Forward Soit $\pi_t(e) = \mathbb{P}(o_1, \dots, o_t, e_t = e)$ la probabilité d'avoir généré la séquence $\mathbf{O} = o_1 \dots o_t$ et d'être arrivé sur l'état e à l'instant t. Cette variable peut être calculée de manière inductive :

Initialisation $\pi_1(e) = \mathbb{P}(e) \cdot \mathbb{P}(o_1 \mid e)$

Induction $\pi_t(e) = \left(\sum_{e' \in \mathbf{E}} \pi_{t-1}(e') \cdot \mathbb{P}(e \mid e') \right) \mathbb{P}(o_t \mid e)$

Connaissant $\pi_n(e)$ pour tous les états e, le calcul de $\mathbb{P}(\mathbf{O})$ est immédiat :

$$(\mathbf{PR}) \qquad \mathbb{P}(\mathbf{O}) = \sum_{e \in E} \pi_n(e) \tag{2.6}$$

Le principe de l'algorithme dit *forward* consiste à calculer la probabilité de générer le premier symbole de la séquence, puis à chaque étape de l'induction on rajoute un symbole et on itère la procédure jusqu'à avoir calculé la probabilité pour la séquence entière. Avec l'algorithme forward, le calcul de $\mathbb{P}(\mathbf{O})$ admet une complexité en $O(nd^2)$. En effet cet algorithme est une spécilisation de l'algorithme de propagation de messages au cas d'un graphe en forme de peigne.

Un algorithme similaire, l'algorithme *backward*, permet de réaliser ce calcul à l'envers.

L'algorithme Backward Soit $\lambda_t(e) = \mathbb{P}(o_{t+1} \dots o_n \mid e_t = e)$ la probabilité de générer la séquence $\mathbf{O} = o_{t+1} \dots o_n$ en partant de l'état e_t.

Initialisation $\lambda_n(e) = 1$

Induction $\lambda_t(e) = \left(\sum_{e' \in \mathbf{E}} \lambda_{t+1}(e') \cdot \mathbb{P}(e' \mid e) \cdot \mathbb{P}(o_{t+1} \mid e') \right)$

L'algorithme admet la même complexité que l'algorithme forward. Il est possible de combiner les deux algorithmes puisque nous avons l'égalité suivante :

$$(\mathbf{MAR}) \qquad \mathbb{P}(X_t = e, \mathbf{O}) = \pi_t(e) \cdot \lambda_t(e) \tag{2.7}$$

2.3.2.2 Algorithme de Viterbi

Cet algorithme est utilisé pour résoudre notre problème MPE adapté aux HMM. A partir d'une séquence de symboles $\mathbf{O} = o_1, \dots, o_n$, le but est de trouver la séquence d'états qui a la probabilité maximale de générer $\mathbf{O}$. Ce qui importe, ce n'est pas tant la probabilité mais plutôt l'affectation des variables cachées. L'algorithme de Viterbi est un algorithme de programmation dynamique très similaire à l'algorithme Forward et permet de résoudre efficacement ce problème.

Soit $\delta_t(e) = \max_{e_1 \dots e_{t-1}} \mathbb{P}(e_1 \dots e_t = s, o_1 \dots o_t)$ la probabilité maximale de générer la séquence $\mathbf{O} = o_1 \dots o_t$ suivant un unique chemin arrivant à l'état e à l'instant t. De la même manière que pour $\pi_t(e)$ dans l'algorithme forward nous avons :

Initialisation $\delta_1(e) = \mathbb{P}(e) \cdot \mathbb{P}(o_1 \mid e)$

Induction $\delta_t(e) = \left(\max_{e' \in \mathbf{E}} \delta_{t-1}(e') \cdot \mathbb{P}(e \mid e') \right) \mathbb{P}(o_t \mid e)$

Connaissant $\delta_n(e)$ pour tous les états e nous pouvons calculer la probabilité maximale suivant un chemin C par :

$$\mathbb{P}(\mathbf{O} \mid C) = \max_{e \in \mathbf{E}} \delta_n(e)$$

Mais ce qui importe, dans cet algorithme ce n'est pas cette probabilité mais plutôt l'affectation des variables qui a permis de l'obtenir. Il suffit alors à chaque étape t de l'induction et pour chaque état e de mémoriser l'état $e' = \psi_t(e)$ qui maximise l'équation d'induction.

Initialisation $\psi_1(e) = 0$

Induction $\psi_t(e) = \left(\operatorname*{argmax}_{e' \in \mathbf{E}} \delta_{t-1}(e') \cdot \mathbb{P}(e \mid e') \right)$

Après le calcul des δ et ψ, il ne reste plus qu'à dérouler le chemin $e_1^* \dots e_n^*$ de probabilité maximale à l'aide de l'induction suivante :

Initialisation $e_n^* = \operatorname*{argmax}_{e \in \mathbf{E}} \delta_n(e)$

Induction $e_t^* = \psi_t(e_{t+1}^*)$

La différence principale entre l'algorithme de Viterbi et l'algorithme Forward résulte dans la dernière induction et dans la maximisation des probabilités plutôt que leur somme.

Les réseaux de fonctions de coûts 3

Les réseaux de fonctions de coûts sont une extension des réseaux de contraintes qui permet de modéliser des préférences sur certaines combinaisons de valeurs ou le coût d'une violation de contrainte. Ces techniques offrent des algorithmes performants pour rechercher des solutions minimisant le coût des violations de contraintes.

Dans un premier temps nous définissons rapidement le formalisme des Csp pour en arriver à celui des Wcsp. Ensuite seront présentées les diverses techniques permettant de résoudre ces problèmes d'optimisation. La dernière section présente comment avec les Wcsp nous pouvons modéliser les réseaux bayésiens.

3.1 Formalismes et définitions

3.1.1 Réseau de contraintes

Définition 3.1 Réseau de contraintes [Montanari, 1974]
Un réseau de contraintes ou Csp (Constraint Satisfaction Problem) est un triplet $(\mathbf{X}, \mathbf{D}, \mathbf{C})$ avec $\mathbf{X}$ un ensemble de n variables, $\mathbf{D} = \{D_{X_1}, \ldots, D_{X_n}\}$ l'ensemble des domaines finis pour chaque variable et $\mathbf{C}$ un ensemble de m contraintes. Chaque $c_{\mathbf{S}} \in \mathbf{C}$ porte sur un sous-ensemble de variables $\mathbf{S} \subseteq \mathbf{X}$, appelé *portée* de la contrainte. Elle n'autorise qu'une partie des combinaisons de valeurs (appelés *tuples*) possibles pour les variables de $\mathbf{S}$.

L'*arité* d'une contrainte est le nombre de variables sur lesquelles elle porte (c'est-à-dire $\mid \mathbf{S} \mid$ pour la contrainte $c_{\mathbf{S}}$). On appelle contrainte *unaire*, *binaire*, voire *ternaire* et *quaternaire*, une contrainte portant respectivement sur une, deux, trois et quatre variables. Pour les contraintes d'arité plus grande, nous parlerons généralement de contraintes *n-aires*.

Définition 3.2 Affectations
L'*affectation* d'une variable $X_i \in \mathbf{X}$ est une paire (X_i, v_i), $v_i \in D_{X_i}$, habituellement notée $\{X_i \leftarrow v_i\}$.

Une affectation de l'ensemble $\mathbf{X}$ est l'union des affectations des variables de $\mathbf{X}$. Une affectation *complète* est l'affectation de toutes les variables du problème, tandis qu'une affectation *partielle* ne porte pas forcément sur toutes les variables. Une affectation *cohérente* est une affectation qui ne viole aucune contrainte et une affectation complète et cohérente est appelée une *solution* du problème.

Définition 3.3 Projection d'une affectation
La *projection* d'une affectation (partielle ou totale) $\mathcal{A} = \{X_1 \leftarrow v_1, \ldots, X_k \leftarrow v_k\}$ sur l'ensemble $\mathbf{Y} = \{X_{i_1}, \ldots, X_{i_p}\} \subset \{X_1, \ldots, X_k\}$ est l'affectation partielle $\{X_{i_1} \leftarrow v_{i_1}, \ldots, X_{i_p} \leftarrow v_{i_p}\}$. Elle est notée $\mathcal{A}[\mathbf{Y}]$.

On notera $\mathbf{X}_{\mathcal{A}}$ l'ensemble des variables affectées dans $\mathcal{A}$.

Définition 3.4 Extension d'une affectation
L'*extension* d'une affectation $\mathcal{A}$ par une affectation $\mathcal{A}'$, telle que $\mathbf{X}_{\mathcal{A}} \cap \mathbf{X}_{\mathcal{A}'} = \emptyset$ est l'affectation $\mathcal{A} \cup \mathcal{A}'$.

Problème de satisfaction de contraintes
Entrée : Une instance Csp $\mathcal{P}$
Sortie : l'instance $\mathcal{P}$ admet-elle au moins une solution ?
Le problème de satisfaction de contraintes consiste à donner une valeur à chaque variable de telle sorte que toutes les contraintes soient satisfaites. C'est un problème NP-Complet.

Nous pouvons représenter la structure du Csp sous forme graphique, appelée graphe de contraintes.

Définition 3.5 Graphe de contraintes
Soit un Csp $(\mathbf{X}, \mathbf{D}, \mathbf{C})$, le graphe $G = (\mathbf{X}, \mathbf{A})$ est le graphe de contraintes associé à $\mathcal{P}$ tel que $\{X_i, X_j\} \in \mathbf{A}$ s'il existe une contrainte $c_{\mathbf{S}}$ avec $\{X_i, X_j\} \subseteq \mathbf{S}$.

Remarque 3.6 *Dans le cas d'un* Csp *binaire*[1] *le graphe de contraintes est simplement* $G = (\mathbf{X}, \mathbf{C})$. *Toutes les notions de voisinage, de degré, présentées dans les graphes sont transposées dans les* Csp *au travers de leur graphe de contraintes. Par exemple, l'ensemble des voisins d'une variable* X*, correspondant à l'ensemble des variables partageant au moins une contrainte avec* X*, est exactement identifié par l'ensemble* $\texttt{voisins}(X)$ *dans le graphe de contraintes associé.*

Exemple 3.7 *Pour illuster ces définitions nous considérons le* Csp $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{C})$ *avec :*
- $\mathbf{X} = \{A, B, C, D, E, F, G, H\}$
- $\mathbf{D} = \{D_A, D_B, D_C, D_D, D_E, D_F, D_G, D_H\}$ *tel que* $D_C = D_D = D_G = D_H = \{a, b, c\}$ *et* $D_A = D_B = D_E = D_F = \{a, b\}$.
- $\mathbf{C} = \{c_{ABC}, c_{AD}, c_{BE}, c_{EF}, c_{EG}, c_{FH}, c_{GH}\}$

La figure 3.1 représente le graphe de contraintes associé à $\mathcal{P}$.

Les combinaisons (tuples) interdites sont notifiées par ✗ *et les combinaisons (tuples) possibles par* ✓. *Les tables des contraintes sont les suivantes :*

1. Le Csp admet des contraintes d'arité au plus binaires

c_{ABC}		C		
		a	b	c
A	B			
a	a	✓	✗	✗
	b	✗	✓	✗
b	a	✓	✓	✓
	b	✗	✗	✓

c_{AD}		D		
		a	b	c
A	a	✗	✓	✓
	b	✓	✗	✓

c_{BE}		E	
		a	b
B	a	✓	✓
	b	✗	✓

c_{EF}		F	
		a	b
E	a	✗	✓
	b	✓	✗

c_{EG}		G		
		a	b	c
E	a	✗	✓	✓
	b	✗	✗	✓

c_{FH}		H		
		a	b	c
F	a	✓	✓	✗
	b	✓	✗	✓

c_{GH}		H		
		a	b	c
G	a	✗	✓	✓
	b	✓	✗	✓
	c	✓	✓	✗

Par exemple, la contrainte c_{BE} interdit le tuple (b, a) et autorise les autres. Donc l'affectation $\{B \leftarrow b, E \leftarrow a\}$ viole la contrainte c_{BE}. L'affectation partielle $\mathcal{A} = \{A \leftarrow a, B \leftarrow b, C \leftarrow b, G \leftarrow a, H \leftarrow c\}$ est une affectation cohérente, car elle ne viole aucune contrainte. En effet même si nous remarquons que la contrainte c_{EG} ne pourra pas être satisfaite la contrainte n'est pas considérée comme violée car la variable E n'est pas affectée dans $\mathcal{A}$. Le Csp *admet au moins une solution dont l'affectation $\{A \leftarrow a, B \leftarrow a, C \leftarrow a, D \leftarrow c, E \leftarrow b, F \leftarrow a, G \leftarrow c, H \leftarrow a\}$.*

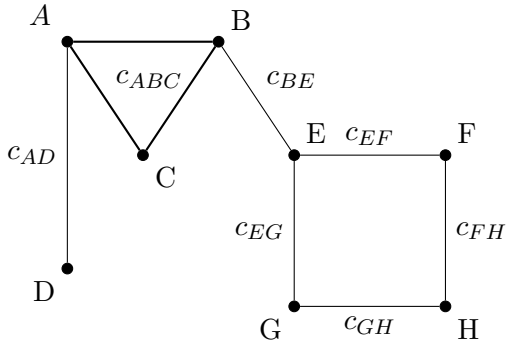


Figure 3.1 – Graphe de contraintes associé au Wcsp de l'exemple 3.7

3.1.2 Réseaux de fonctions de coûts

Les Csp utilisent seulement des contraintes pour modéliser l'autorisation ou l'interdiction de combinaisons de valeurs. Pour modéliser des préférences pour certains tuples de valeurs plusieurs formalismes ont étendu les Csp, dont les réseaux de contraintes valuées (Vcsp, *Valued Constraint Satisfaction Problem*) [Bistarelli et al., 1999]. Tous ces formalismes ajoutent aux Csp une structure de valuation permettant de définir une structure algébrique caractérisant les coûts associés aux solutions.

Définition 3.8 Structure de valuation
Une *structure de valuation* est un quintuplet $\mathcal{S} = (\mathbf{V}, \oplus, \prec, \bot, \top)$ où $\mathbf{V}$ est l'ensemble des valuations, $\oplus$ l'opérateur de combinaison dans $\mathbf{V}$, $\prec$ défini un ordre total sur $\mathbf{V}$ dont $\bot$ est le minimum et $\top$ le maximum. L'opérateur $\oplus$ doit être commutatif, associatif et monotone sur $\mathbf{V}$, $\bot$ doit être son élément neutre et $\top$ un élément absorbant.

Pour les réseaux de fonctions de coûts (Wcsp, *Weighted Constraint Satisfaction Problem*) la structure de valuation est la suivante.

Définition 3.9 Structure de valuation des Wcsp
La *structure de valuation* est un quintuplet $\mathcal{S} = (E^+, \oplus, \leqslant, 0, \top)$ où $E^+ = [0 \dots \top]$ avec $\top \in \mathbb{N} \cup \{+\infty\}$, le plus grand coût, il représente une incohérence, un coût interdit. $\oplus$ est défini comme une somme bornée dans E^+ :

$$\forall a, b \in E^+,\ a \oplus b = \min\{\top, a + b\}$$

Un opérateur $\ominus$ est également défini tel que

$$\forall a \geqslant b \in E^+,\ a \ominus b = \begin{cases} \top & \text{si } a = \top \\ a - b & \text{sinon.} \end{cases}$$

Définition 3.10 Réseau de fonctions de coûts
Un réseau de fonctions de coûts,Wcsp, est un triplet $(\mathbf{X}, \mathbf{D}, \mathbf{F})$ associé à la structure de valuation $(\mathbf{E}^+, \oplus, \leqslant, 0, \top)$ avec $\mathbf{X} = \{X_1, \dots, X_n\}$ un ensemble de variables, $\mathbf{D} = \{D_1, \dots, D_n\}$ un ensemble de domaines finis et $\mathbf{F} = \{f_{\mathbf{S}_1}, \dots, f_{\mathbf{S}_m}\}$ un ensemble de fonctions de coûts avec $\forall 1 \leqslant i \leqslant m$, $\mathbf{S}_i \subset \mathbf{X}$. Une fonction de coûts $f_{\mathbf{S}_i} : \left(\underset{X_k \in \mathbf{S}_i}{\times} D_{X_k} \right) \to E^+$

Il existe une fonction de coûts particulière associée au Wcsp d'arité nulle $f_\emptyset$, c'est une constante qui est ajoutée à n'importe quelle affectation.

Définition 3.11 Coût d'une affectation
Le coût d'une affectation complète $\mathcal{A} = \{X_1 \leftarrow x_1, \dots, X_n \leftarrow x_n\}$ est donné par

$$cout(\mathcal{A}) = f_\emptyset \bigoplus_{i=1}^{m} f_{\mathbf{S}_i}(\mathcal{A}[\mathbf{S}_i])$$

Une affectation $\mathcal{A}$ est cohérente si $cout(\mathcal{A}) < \top$. Notons qu'avec $\top = 1$, le Wcsp se réduit à un Csp : un coût de 0 signifie la satisfaction et 1 la violation d'au moins une contrainte.

Problème d'optimisation

Entrée : Une instance Wcsp $\mathcal{P}$

Sortie : quel est le coût minimal c d'une solution de $\mathcal{P}$?

$$c = \min_{\mathcal{A} \in D_{X_1} \times \dots \times D_{X_n}} cout(\mathcal{A})$$

Exemple 3.12 *Nous considérons le* WCSP $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{F})$ *et* $\top = 100$ *avec :*

- $\mathbf{X} = \{A, B, C, D\}$
- $\mathbf{D} = \{D_A, D_B, D_C, D_D\}$ *avec* $D_C = \{a, b, c\}$ *et* $D_A = D_B = D_D = \{a, b\}$.
- $\mathbf{F} = \{f_{AB}, f_{BC}, f_{BCD}, f_B\}$ *avec la définition des fonctions suivantes :*

f_{BCD}		C		
		a	b	c
B	D			
a	a	3	2	6
	b	1	5	5
b	a	0	12	**0**
	b	1	4	2

f_{AB}		B	
		a	b
A	a	3	**2**
	b	0	5

f_{AC}		C		
		a	b	c
A	a	2	4	**1**
	b	6	4	5

f_B	B	
	a	b
	3	**2**

Le coût minimal c *de ce problème est* 5 *et une affectation possible avec ce coût est* $\mathcal{A} = \{A \leftarrow a, B \leftarrow b, C \leftarrow c, D \leftarrow a\}$*, en effet nous avons bien* $f_{BDC}(b, a, c) \oplus f_{AB}(a, b) \oplus f_{BC}(b, c) \oplus f_B(b) = 0 \oplus 2 \oplus 1 \oplus 2 = 5$.

Les trois opérations suivantes sur les fonctions de coûts permettent de réaliser toutes les opérations qui seront présentées dans la section suivante.

Définition 3.13 Conditionnement
Soit une fonction f définie sur un ensemble $\mathbf{S}$. Le *conditionnement* de f par rapport à l'affectation d'une variable X_i avec l'une de ses valeurs x_i de son domaine, noté $f(\mathcal{A} \cup \{X_i \leftarrow x_i\})$, est défini par une nouvelle fonction g sur l'ensemble $\mathbf{S} \setminus \{X_i\}$ telle que

$$\forall \mathcal{A} \in D_{\mathbf{S} \setminus \{X_i\}}\, g(\mathcal{A}) = f(\mathcal{A} \cup \{X_i \leftarrow x_i\})$$

Définition 3.14 Addition de deux fonctions
Soient deux ensembles de variables $\mathbf{S}_1$ et $\mathbf{S}_2$ et deux fonctions f_1 et f_2 définies respectivement sur $\mathbf{S}_1$ et $\mathbf{S}_2$. La combinaison des deux fonctions $(f_1 \oplus f_2)$ définie sur l'ensemble $\mathbf{S}_1 \cup \mathbf{S}_2$ est définie par

$$\forall \mathcal{A} \in D_{\mathbf{S}_1 \cup \mathbf{S}_2},\ (f_1 \oplus f_2)(\mathcal{A}) = f_1(\mathcal{A}[\mathbf{S}_1]) \oplus f_2(\mathcal{A}[\mathbf{S}_2])$$

Remarque 3.15 *Il est également possible de réaliser une soustraction de deux fonctions, si* $\mathbf{S}_2 \subseteq \mathbf{S}_1$*. La différence des deux fonctions* $(f_1 \ominus f_2)$ *est définie par*

$$\forall \mathcal{A} \in \mathbf{S}_1,\ (f_1 \ominus f_2)(\mathcal{A}) = f_1(\mathcal{A}) \ominus f_2(\mathcal{A}[\mathbf{S}_2])$$

si $\forall \mathcal{A} \in \mathbf{S}_1,\ f_1(\mathcal{A}) \geqslant f_2(\mathcal{A}[\mathbf{S}_2])$.

Définition 3.16 Projection d'une fonction
Soient deux ensembles de variables $\mathbf{S}$ et $\mathbf{S}'$ tels que $\mathbf{S}' \subset \mathbf{S}$ et une fonction f définie sur l'ensemble $\mathbf{S}$. La *projection de la fonction* f sur l'ensemble $\mathbf{S}'$, notée $f[\mathbf{S}']$ est définie par

$$\forall \mathcal{B} \in D_{\mathbf{S}'},\ f[\mathbf{S}'](\mathcal{B}) = \min_{\substack{\mathcal{A} \in D_{\mathbf{S}}, \\ \mathcal{A}[\mathbf{S}'] = \mathcal{B}}} f(\mathcal{A})$$

Exemple 3.17 *Reprenons le* Wcsp *de l'exemple précédent (3.12 page précédente).*

Le conditionnement *de la fonction f_{AB} par l'affectation $\{A \leftarrow a\}$ crée la fonction unaire $f_{AB}(\{A \leftarrow a\}) = g_B$. $g_B(a) = f_{AB}(a,a)$ et $g_B(b) = f_{AB}(a,b)$. La table ci-contre illustre ce conditionnement.*

$f_{AB}(\{A \leftarrow a\})$	B	
	a	b
	3	2

La projection *de la fonction binaire f_{AB} sur la variable B permet d'obtenir la fonction unaire $f_{AB}[B]$ ci-contre.*
$f_{AB}[B](a) = \min\{f_{AB}(a,a), f_{AB}(b,a)\} = \min\{3,0\} = 0$
$f_{AB}[B](b) = \min\{f_{AB}(a,b), f_{AB}(b,b)\} = \min\{2,5\} = 2$

$f_{AB}[B]$	B	
	a	b
	0	2

La somme *des deux fonctions f_{AB} et f_{AC} est donnée par la table ci-contre.*
Par exemple, le calcul pour obtenir le coût du tuple (a,b,c) est $(f_{AB} \oplus f_{BC})(a,b,c) = f_{AB}(a,b) \oplus f_{AC}(a,c) = 2 \oplus 1 = 3$

$f_{AB} \oplus f_{AC}$		C		
		a	b	c
A	B			
a	a	5	7	4
	b	4	6	3
b	a	6	4	5
	b	11	9	10

3.2 Techniques d'optimisation

3.2.1 Élimination de variables

Une première approche pour la résolution du problème Wcsp consiste à adapter les algorithmes d'élimination de variables [Dechter, 1999] prévus pour les Csp.

La définition suivante présente le processus utilisé pour éliminer une variable du Wcsp.

Définition 3.18 Élimination d'une variable
Soit le Wcsp $(\mathbf{X}, \mathbf{D}, \mathbf{F})$. L'élimination d'une variable $X_i \in \mathbf{X}$ s'effectue en trois étapes :

1. Identification de toutes les fonctions de coûts portant sur la variable X_i.
2. Ajout au réseau d'une fonction f, portant sur les variables $X_i \cup \texttt{voisins}(X_i)$, résultant de la somme de toutes les fonctions identifiées en 1.
3. Projection de la fonction f sur l'ensemble $\texttt{voisins}(X_i)$.

La variable X_i et les fonctions portant sur elle sont alors supprimées. La variable X_i a été éliminée.

L'algorithme d'élimination de variables `VE` (*Variable Elimination*) dans les WCSP consiste à réduire itérativement le nombre de variables du problème WCSP en préservant le coût de la solution optimale.

A la fin de l'algorithme il ne reste plus aucune variable, seule la fonction de coûts de portée vide ($f_\emptyset$) subsiste et est égale à l'optimum du problème.

L'efficacité de cet algorithme dépend fortement de l'ordre dans lequel les variables sont éliminées. En effet chaque élimination de variable crée une nouvelle fonction de coûts dont l'arité correspond au nombre de variables voisines de celle éliminée dans le réseau modifié. La complexité est exponentielle par rapport au nombre (maximum) de variables voisines d'une variable éliminée. A noter que trouver un ordre optimal est NP-dur et que la complexité de `VE` en temps et en mémoire est exponentielle par rapport à la largeur d'arbre.

Exemple 3.19 *Reprenons le* WCSP *de l'exemple 3.12. Soit l'ordre d'élimination [A B C D].*

1. *Elimination de la variable A :*
   a) *Les fonctions portant sur A sont f_{AB} et f_{AC}.*
   b) *La table de la fonction $f_{ABC} = f_{AB} \oplus f_{AC}$ est donnée dans l'exemple précédent.*
   c) *La projection de la fonction f_{ABC} sur $\{BC\}$ est donnée par la table qui suit. Le problème est alors constitué des fonctions de coûts f_{BCD}, f_{BC}, f_B.*

f_{BC}		C		
		a	b	c
B	a	2	4	4
	b	4	6	3

2. *Elimination de la variable B :*
   a) *Les fonctions portant sur B sont f_{BCD}, f_{BC} et f_B.*
   b) *Les valeurs de la fonction $f'_{BCD} = f_{BCD} \oplus f_{BC} \oplus f_B$ sont données par la table qui suit.*
   c) *La projection de la fonction f'_{BCD} sur $\{CD\}$ donnée par la table f_{CD}. Seule la fonction f_{CD} est encore présente dans le réseau*

f'_{BCD}		C		
B	D	a	b	c
a	a	8	9	12
	b	6	12	12
b	a	5	20	5
	b	7	12	7

f_{DC}		C		
		a	b	c
D	a	5	9	5
	b	6	12	7

3. *Elimination de la variable C : Il ne reste plus que la fonction f_{CD} il suffit de faire la projection de la fonction sur $\{D\}$ dont le résultat est la table suivante*

	B	
f_D	a	b
	5	6

4. Elimination de la variable D : Même principe que pour la variable C. La fonction f_D est projetée sur l'ensemble vide d'où nous obtenons :

$f_\emptyset$	5

3.2.2 Recherche arborescente

Pour trouver une affectation des variables d'un problème WCSP de coût minimum, une approche courante est d'utiliser un algorithme de recherche arborescente en profondeur d'abord : l'algorithme d'évaluation et de séparation ou DFBB (*Depth First Branch and Bound*). DFBB est décrit par l'algorithme 1, le premier appel à DFBB est DFBB($\mathbf{X}, \emptyset, \top$). A chaque nœud de l'arbre de recherche (correspondant à l'affectation d'une variable à une valeur), on calcule un minorant $lb(\mathcal{A})$ du coût des solutions contenues dans le sous-arbre courant (correspondant à toutes les extensions possibles de l'affectation courante $\mathcal{A}$) ainsi qu'un majorant ub du coût de ces solutions. Ce majorant est donné par la valeur minimum des coûts des solutions déjà trouvées. Si le minorant est plus grand que le majorant alors toutes les extensions de l'affectation courante ne peuvent pas être des solutions optimales, il n'est pas nécessaire de continuer la recherche à partir de cette affectation.

Algorithme 1: DFBB($\mathbf{Y}, \mathcal{A}, ub$) : coût

Entrée :
un ensemble de variables $\mathbf{Y}$
une affectation $\mathcal{A}$
une valeur $\in E^+$

Pré-conditions :
$\mathbf{Y}$ contient toutes les variables de $\mathbf{X}$ qui ne sont pas encore affectées dans $\mathcal{A}$, *i.e.* $\mathbf{Y} = \mathbf{X} \setminus \mathbf{X}_\mathcal{A}$

Post-relations :
renvoie $\min(ub, \{cout(\mathcal{A}') \mid \mathcal{A}'$ est une affectation complète telle que $\mathcal{A} \subseteq \mathcal{A}'\})$

$c \leftarrow lb(\mathcal{A})$;
if $c < ub$ **then**
  if $\mid \mathcal{A} \mid = n$ **then return** c;
  Choisir X dans $\mathbf{Y}$;
  $\mathbf{Y} \leftarrow \mathbf{Y} \setminus \{X\}$;
  forall the $x \in D_X$ **do**
    $ub \leftarrow \min(ub, \text{DFBB}(\mathbf{Y}, \mathcal{A} \cup \{X \leftarrow x\}, ub)$;
  return ub;
return $\top$;

Exemple 3.20 *Reprenons le* Wcsp *de l'exemple 3.12 avec* $\top = 20$ *(*$ub = \top$*). La figure 3.2 représente l'arbre de recherche de l'algorithme* Dfbb, le calcul de $lb(\mathcal{A})$ est le coût des fonctions dont toutes les variables sont affectées dans l'affectation courante $\mathcal{A}$, dans l'arbre de recherche la valeur de $lb(\mathcal{A})$ est donnée entre parenthèse, en bleu lorsque $lb(\mathcal{A}) < ub$ et en rouge lorsque $lb(\mathcal{A}) \geqslant ub$. Par exemple pour l'affectation $\mathcal{A} = \{A \leftarrow a, B \leftarrow a\}$, la valeur de lb est donnée par $f_{AB}(a,a) + f_B(a) = 3 \oplus 3$. L'affectation complète $\mathcal{A} = \{A \leftarrow a, B \leftarrow a, C \leftarrow a, D \leftarrow a\}$ admet un coût inférieur à $ub = \top$ donc la meilleure solution a un coût inférieur ou égal à 11, la nouvelle valeur de ub. L'affectation partielle $\mathcal{A}_1 = \{A \leftarrow a, B \leftarrow b\}$ a un coût de 10 qui est plus grand que la valeur courante de ub égale à 9, il n'existe donc pas d'extension de $\mathcal{A}_1$ sur C et D ayant un coût inférieur à ub, l'algorithme arrête la recherche pour cette affectation.

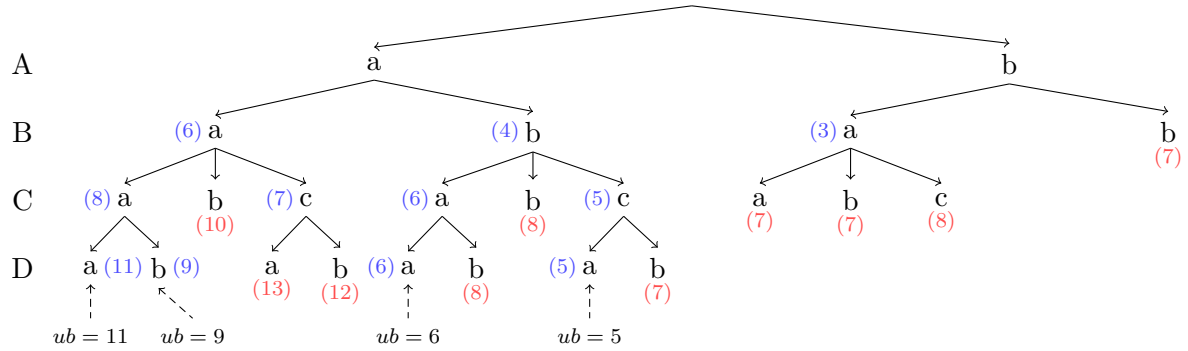


Figure 3.2 – Arbre de recherche de Dfbb

La complexité temporelle de Dfbb est en $O(md^n)$ où d est la taille maximale du domaine des variables et sa complexité en mémoire est en $O(n)$.

Cependant, l'efficacité de l'algorithme dépend de la qualité du minorant $lb(\mathcal{A})$ utilisé. Un minorant simple peut être calculé en prenant le coût des fonctions dont toutes les variables sont affectées dans l'affectation courante. Mais ce minorant n'est pas de bonne qualité.

Dans la section 3.2.3 nous allons voir des méthodes de filtrage permettant d'obtenir de meilleurs minorants contenus dans la fonction $f_\emptyset$.

3.2.3 Cohérences locales

Durant ces dernières années, des minorants de qualité croissante ont été définis étendant les propriétés des cohérences locales aux Wcsp à l'aide de méthodes de filtrage. Un algorithme de filtrage transforme un problème Wcsp en un problème Wcsp équivalent. Son but principal étant de permettre la déduction de valeurs impossibles, d'augmenter le coût de la fonction constante $f_\emptyset$ qui servira de minorant [2] et aussi de déduire des coûts unaires pour guider les heuristiques de choix de valeurs et de variables.

2. d'où la nécessité d'avoir des coûts positifs

Définition 3.21 Équivalence de Wcsp [Cooper and Schiex, 2004]
Deux Wcsp, définis sur un même ensemble de variables, sont *équivalents* s'ils ont le même ensemble de solutions optimales et la même distribution des coûts pour toutes leurs affectations.

3.2.3.1 Opérations sur les réseaux de fonctions de coûts

Les reformulations préservant l'équivalence Ept (*Equivalence Preserving Transformations*) des Wcsp agissent en déplaçant les coûts entre les fonctions. Leur objectif est de transférer les coûts vers les fonctions unaires ou la fonction constante (la fonction $f_\emptyset$). Les opérations de reformulation sont : la projection et l'extension d'un coût.

Définition 3.22 Transformation par projection
Une *transformation par projection* est le déplacement d'un coût α d'une fonction $f_{\mathbf{S}}$ de portée $\mathbf{S}$ vers une fonction unaire $f_X(x)$, où $X \in \mathbf{X}$ et $x \in D_X$.

Pour conserver des coûts positifs, le coût projeté ne doit pas être supérieur au minimum des coûts concernés par cette projection (c'est-à-dire $\alpha \in [1, \min_{\mathcal{A}[X]=x} f_{\mathbf{S}}(\mathcal{A})]$). L'algorithme suivant décrit les opérations nécessaires pour réaliser la projection d'un coût d'une fonction $f_{\mathbf{S}}$ vers $f_X(x)$.

Algorithme 2: projection$(f_{\mathbf{S}}, X, x, \alpha)$

Entrée :
une fonction de coût sur l'ensemble de variable $\mathbf{S}$ $f_{\mathbf{S}}$
une variable $X \in \mathbf{S}$
une valeur $x \in D_X$
une valeur $\alpha \in E^+$

Pré-conditions :
$\alpha \in [1, \min_{\mathcal{A}[X]=x} f_{\mathbf{S}}(\mathcal{A})]$

forall the $\mathcal{A}$ ***tel que*** $\mathcal{A}[X] = x$ **do**
  $f_{\mathbf{S}}(\mathcal{A}) \leftarrow f_{\mathbf{S}}(\mathcal{A}) \ominus \alpha$;
$f_X(x) \leftarrow f_X(x) \oplus \alpha$;

Définition 3.23 Transformation par extension
Une *transformation par extension* est le déplacement d'un coût α de la fonction $f_X(x)$ vers une fonction $f_{\mathbf{S}}$ de portée $\mathbf{S}$ avec $X \in \mathbf{S}$.

Pour conserver des coûts positifs, le coût à étendre ne doit pas être supérieur à $f_X(x)$ (c'est-à-dire $\alpha \in [1, f_X(x)]$). L'algorithme suivant décrit les opérations nécessaires pour réaliser cette extension.

Remarque 3.24 *La projection d'un coût vers la fonction constante $f_\emptyset$ se fait par* `projection`$(f_{\mathbf{S}}, \emptyset, \emptyset, \alpha)$ *ainsi* $\min_{\mathcal{A}[\emptyset]=\emptyset} f_{\mathbf{S}}(\mathcal{A})$ *devient* $\min_{\mathcal{A}} f_{\mathbf{S}}(\mathcal{A})$.

Algorithme 3: extension($f_{\mathbf{S}}, X, x, \alpha$)

Entrée :
une fonction de coût sur l'ensemble de variable $\mathbf{S}$ $f_{\mathbf{S}}$
une variable $X \in \mathbf{S}$
une valeur $x \in D_X$
une valeur $\alpha \in E^+$

Pré-conditions :
$\alpha \in [1, f_X(x)]$

---

forall the $\mathcal{A}$ ***tel que*** $\mathcal{A}[X] = x$ **do**
  $f_{\mathbf{S}}(\mathcal{A}) \leftarrow f_{\mathbf{S}}(\mathcal{A}) \oplus \alpha$;
$f_X(x) \leftarrow f_X(x) \ominus \alpha$;

Pour illustrer ces opérations et les cohérences locales, nous définirons des WCSP binaires $(\mathbf{X}, \mathbf{D}, \mathbf{F})$ que nous représenterons par un graphe étiqueté où les sommets représentent les valeurs et pour chaque paire de variables $X, Y \in \mathbf{X}$ où $f_{XY} \in \mathbf{F}$, pour chaque valeur $x \in D_X$ et $y \in D_Y$ tel que $f_{XY}(x, y) > 0$, une arête connecte les sommets associés aux valeurs $\{X \leftarrow x\}$ et $\{Y \leftarrow y\}$. L'étiquette de cette arête est la valeur du coût $f_{XY}(x, y)$. Les coûts des fonctions unaires sont représentés par l'étiquette associée aux sommets. Ce graphe est différent du graphe de contraintes, il s'apparente à la microstructure d'un CSP.

Exemple 3.25 *Soit un* WCSP *de deux variables* A, B *de domaine* $\{a, b\}$. *La figure 3.3a représente le* WCSP *sous forme de graphe telle que nous ayons deux fonctions unaires* f_A *et* f_B *avec* $f_A(a) = f_B(a) = 1$, *et l'arête qui relie les valeurs* b *des variables* A *et* B *représente un coût de* 1 *pour la fonction binaire* f_{AB} *:* $f_{AB}(b, b) = 1$. *La figure 3.3b représente le* WCSP *équivalent après une transformation par extension du coût* $f_B(a)$ *vers la fonction binaire* f_{AB}.

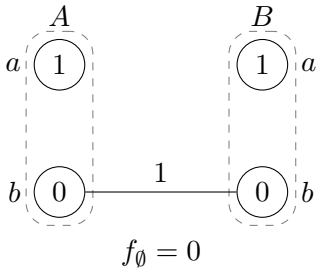
- $f_{AB}(a, a) \oplus f_B(a) = 0 \oplus 1$ *devient la nouvelle valeur de* $f_{AB}(a, a)$
- $f_{AB}(b, a) \oplus f_B(a) = 0 \oplus 1$ *devient la nouvelle valeur de* $f_{AB}(b, a)$
- $f_B(a) \ominus f_B(a) = 1 \ominus 1 = 0$ *devient la nouvelle valeur de* $f_B(a)$

La figure 3.3c représente le WCSP *équivalent après la transformation par projection du coût* 1 *de la* f_{AB} *vers* $f_A(b)$.

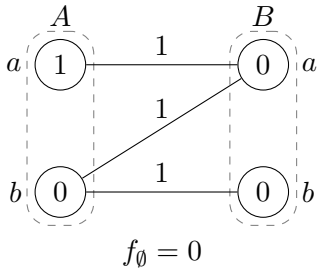
- $f_A(b) \oplus 1 = 0 \oplus 1 = 1$ *devient la nouvelle valeur de* $f_A(b)$
- $f_{AB}(b, a) \ominus 1 = 1 \oplus 1 = 0$ *devient la nouvelle valeur de* $f_{AB}(b, a)$
- $f_{AB}(b, b) \ominus 1 = 1 \oplus 1 = 0$ *devient la nouvelle valeur de* $f_{AB}(b, b)$

Enfin `projection`$(f_{\mathbf{S}}, \emptyset, \emptyset, \alpha)$ *donnée par la figure 3.3d produit un minorant* $f_\emptyset = 1$.

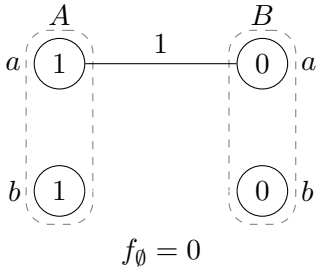
Pour la suite de la section nous définissons un WCSP $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{F})$ et X_i, X_j deux de ses variables.



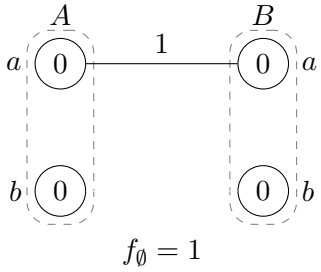
(a) WCSP d'origine



(b) WCSP après extension du coût $f_B(a)$ vers f_{AB}



(c) WCSP après projection depuis f_{AB} vers $f_A(b)$



(d) WCSP après projection depuis f_A vers $f_\emptyset$

Figure 3.3 – Opérations d'extension et de projections

3.2.3.2 Cohérence de nœud

La cohérence de nœud (NC, *Node Consistency*) est la plus simple des propriétés que l'on puisse établir sur $\mathcal{P}$ en projetant tous les coûts unaires vers la fonction $f_\emptyset$ [Larrosa, 2002].

Définition 3.26 Cohérence de nœud (NC)
Une variable X_i est *nœud cohérente* si

1. $\exists x_i \in D_{X_i}$ tel que $f_{X_i}(x_i) = 0$
2. $\forall x_i \in D_{X_i}$, $f_{X_i}(x_i) \oplus f_\emptyset < \top$

$\mathcal{P}$ est nœud cohérent si toutes ses variables sont nœud cohérentes.

Si aucune valeur n'a de coût unaire nul alors nous pouvons projeter un coût non nul sur $f_\emptyset$. Lorsqu'une valeur x d'une variable ne satisfait pas le second point de la définition 3.26, x est une valeur impossible (interdite), elle peut être éliminée du domaine de la variable concernée.

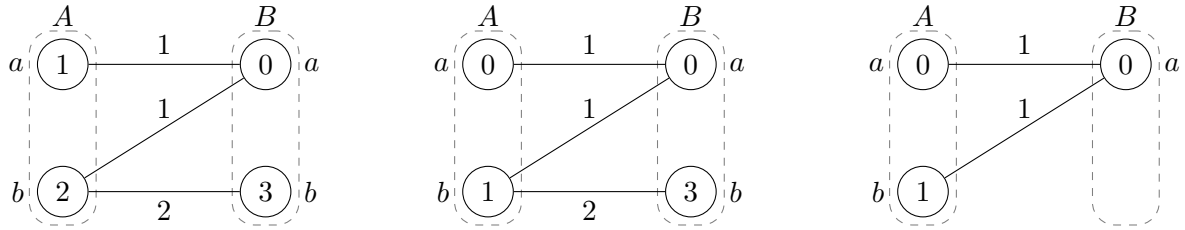
Exemple 3.27 *Soit le* WCSP $\mathcal{P}$ *sur deux variables* A, B *de domaine* $\{a, b\}$*. Avec les fonctions unaires* f_A *et* f_B *et la fonction binaire* f_{AB}*. La figure 3.4a représente* $\mathcal{P}$*.*

La variable A *ne respecte pas la première condition de NC car tous ses coûts unaires sont strictement positifs (coûts égaux à* 1 *et* 2*). La transformation par projection de la fonction unaire* f_A *vers*

la fonction constante $f_\emptyset$ soustrait un coût de 1 de $f_A(a)$ et $f_A(b)$ et ajoute ce coût à $f_\emptyset$ ce qui permet d'obtenir le Wcsp *associé au graphe de la figure 3.4b satisfaisant cette première condition.*

La variable B ne satisfait pas la seconde condition de NC car son affectation à la valeur b nous donne $f_B(b) \oplus f_\emptyset = 3 \oplus 1 = 4 \not< \top = 4$. L'affectation de la variable B à b ($\{B \leftarrow b\}$) ne peut participer à une solution, la valeur b est donc supprimée de son domaine.

Le Wcsp *résultant donné par la figure 3.4c est nœud cohérent.*



(a) Le problème $\mathcal{P}$ d'origine

(b) Établissement de la première propriété de NC

(c) Établissement de NC

Figure 3.4 – Établissement de la cohérence de nœud

Pour établir la cohérence de nœud il existe un algorithme de complexité $O(nd)$ en temps et en espace, où d est la taille maximale des domaines.

3.2.3.3 Cohérence d'arc

Une autre cohérence locale existe, plus forte que la précédente dont l'équivalent dans les Csp est très utilisé : la cohérence d'arc (AC, *Arc Consistency*) [Schiex, 2000].

Définition 3.28 Cohérence d'arc
Une variable X_i est arc-cohérente si

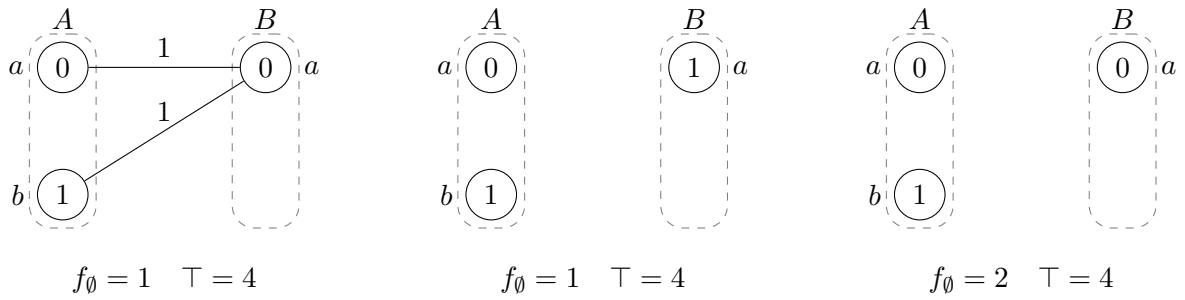
1. $\forall f_{\mathbf{S}} \in \mathbf{F}$ tel que $X_i \in \mathbf{S}$, $\forall x_i \in D_{X_i}$, $f_{\mathbf{S}}[\mathbf{S} \setminus \{X_i\}](x_i) = 0$
2. X_i est nœud cohérente

Le problème $\mathcal{P}$ est arc-cohérent si toutes ses variables sont arc-cohérentes.

Remarque 3.29 *Lorsque le* Wcsp *n'admet que des fonctions de coûts au plus binaires, la première condition peut se réécrire de la manière suivante :*
$\forall X_j \in \texttt{voisins}(X_i)$, $\forall x_i \in D_{X_i}$, $\exists x_j \in D_{X_j}$ *tel que* $f_{X_iX_j}(x_i, x_j) = 0$

Exemple 3.30 *Nous considérons le* Wcsp *de l'exemple 3.27 après l'avoir rendu nœud cohérent (*cf. *figure 3.4c reproduit dans la figure 3.5a). La variable B ne respecte pas la première condition de AC car aucune fonction portant sur sa valeur a n'est nulle. La transformation par projection de la fonction*

f_{AB} vers le coût unaire $f_B(a)$ permet d'obtenir le Wcsp *associé au graphe de la figure 3.5b, qui respecte la première condition de AC. Après cette transformation, la variable B n'est plus nœud cohérente, ne satisfaisant plus la seconde condition de AC. En déplaçant le coût unaire $f_B(a)$ vers $f_\emptyset$, la variable B redevient nœud cohérente, le* Wcsp *résultant associé à la figure 3.5c est arc-cohérent.*



(a) Le problème $\mathcal{P}$ d'origine

(b) Établissement de la première condition de AC

(c) Établissement de la seconde condition de AC

Figure 3.5 – Établissement de la cohérence d'arc

Pour établir la cohérence d'arc, un algorithme présenté par [Larrosa and Schiex, 2003] a une complexité temporelle en $O(md^3 + n^2d^2)$ et une complexité spatiale en $O(md)$.

Cependant suivant l'ordre des transformations de projection utilisé pour rendre le Wcsp arc-cohérent nous pouvons obtenir des Wcsp différents, avec des $f_\emptyset$ différents, mais équivalents. Mais maximiser la valeur de $f_\emptyset$ est NP-dur [Cooper and Schiex, 2004].

Les cohérences suivantes ont été définies pour les Wcsp binaires.

3.2.3.4 Cohérence d'arc complète

Une cohérence plus forte est la cohérence d'arc complète (FAC, *Full Arc Consistency*).

Définition 3.31 Cohérence d'arc complète
Une variables X_i est *complètement arc-cohérente* si

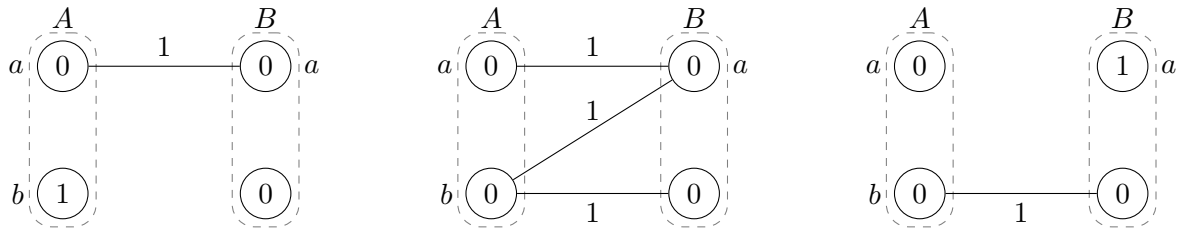
1. $\forall X_j \in \texttt{voisins}(X_i)$, $\forall x_i \in D_{X_i}$, $\exists x_j \in D_{X_j}$, tel que $(f_{X_iX_j} \oplus f_{X_j})[X_i](x_i, x_j) = 0$
2. X_i est nœud cohérente

Le problème $\mathcal{P}$ est *complètement arc-cohérent* si toutes ses variables le sont.

Cependant pour la plupart des problèmes, il n'est pas possible d'établir un Wcsp équivalent complètement arc-cohérent.

Le Wcsp de la figure 3.6a n'admet pas de Wcsp FAC. En effet après l'extension du coût $f_A(b)$ sur f_{AB} représenté par la figure 3.6b et la projection de f_{AB} sur $f_B(a)$ représentée par la figure 3.6c,

la seule transformation possible (extension de $f_B(a)$ sur f_{AB}) oblige à retrouver l'état précédent du WCSP.



(a) Le problème $\mathcal{P}$ d'origine

(b) Transformation par extension de $f_A(b)$

(c) Transformation par projection sur $f_B(a)$

Figure 3.6 – WCSP n'admettant pas de WCSP équivalent FAC

3.2.3.5 Cohérence d'arc directionnelle

La cohérence d'arc directionnelle (DAC, *Directional Arc Consistency*) [Cooper, 2003] n'est basée que sur une partie du voisinage contrairement à FAC. Elle est calculée suivant un ordre sur les variables du problème. Nous considérons un ordre arbitraire sur les variables tel que la variable X_i est préférée à la variable X_j si et seulement si $i < j$.

On note $\texttt{voisins}^+(X_i)$ défini par $\texttt{voisins}(X_i) \cap \{X_j \mid i < j\}$ l'ensemble des voisins de X_i suivants dans l'ordre choisi.

Définition 3.32 Cohérence d'arc directionnelle (DAC)
Une variable X_i est *directionnellement arc-cohérente* si

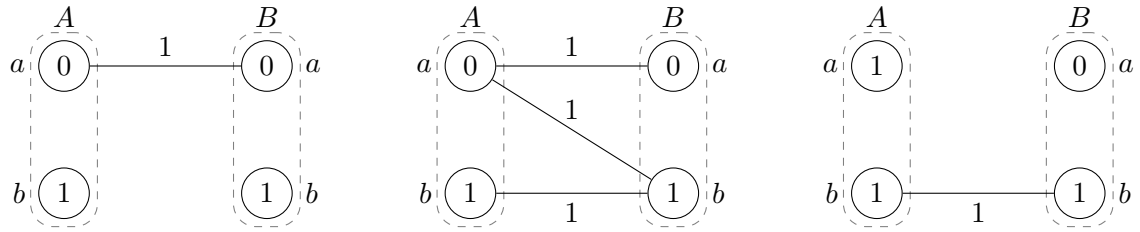
1. $\forall X_j \in \texttt{voisins}^+(X_i)$, $\forall x_i \in D_{X_i}$, $\exists x_i \in D_{X_j}$ tel que $(f_{X_iX_j} \oplus f_{X_j})[X_i](x_i, x_j) = 0$
2. X_i est nœud cohérente

Le problème $\mathcal{P}$ est "directionnellement arc-cohérent" si toutes ses variables le sont.

Exemple 3.33 *Nous considérons le* WCSP $\mathcal{P}$ *sur deux variables* A, B *de domaine* $\{a, b\}$. *Avec les fonctions unaires* f_A *et* f_B *et la fonction binaire* f_{AB} *représentées à la figure 3.7a et l'ordre* $A < B$. *La valeur* a *de la variable* A *ne respecte pas la première condition de DAC car l'affectation* $\{A \leftarrow a\}$ *entraîne un coût de* 1 *quelque soit l'affectation de* B. *Le report de cette information sur la fonction unaire* f_A *(en gardant l'équivalence) se fait en deux étapes :*

- *une transformation par l'extension du coût unaire* $f_B(b)$ *vers la fonction binaire* f_{AB} *(*cf. *figure 3.7b)*
- *une transformation par la projection de la fonction sur le coût unaire* $f_A(a)$. *Le* WCSP *résultant est représenté par la figure 3.7c.*

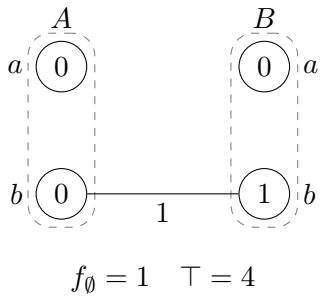
A ce stade, la variable A n'est pas nœud cohérente, mais il suffit de transformer le WCSP *par une projection de la fonction unaire f_A vers $f_\emptyset$. Le* WCSP *résultant (figure 3.7d) est directionnellement arc-cohérent par rapport à l'ordre $A < B$ et a un minorant non nul ($f_\emptyset = 1$).*



(a) Le problème $\mathcal{P}$ d'origine

(b) Transformation par l'extension de $f_B(b)$

(c) Transformation par la projection sur $f_A(a)$



(d) Transformation par la projection sur $f_\emptyset$

Figure 3.7 – Établissement de la cohérence d'arc directionnelle

La complexité de DAC est la même que celle de AC, $O(md^3 + n^2d^2)$ en temps et $O(md)$ en espace.

La cohérence d'arc directionnelle complète (FDAC *Full Directional Arc Consistency*) peut être établie en combinant la cohérence d'arc directionnelle et la cohérence d'arc.

Définition 3.34 Cohérence d'arc directionnelle complète
Une variable X_i est complètement et directionnellement arc-cohérente si

1. X_i est arc-cohérente
2. X_i est directionnellement arc-cohérente

Le problème $\mathcal{P}$ est *complètement et directionnellement arc-cohérent* si toutes ses variables le sont.

3.2.3.6 Cohérence d'arc existentielle

La cohérence d'arc existentielle (EAC *Existential Arc Consistency*) [de Givry et al., 2005b] a pour but (commun aux autres cohérences) d'augmenter la valeur de $f_\emptyset$. Pour cela elle détecte pour chaque

variable les valeurs de coût unaire nul pouvant être augmentées par des transformations de projection et d'extension.

Définition 3.35 Cohérence d'arc existentielle
Une variable X_i est existentiellement arc-cohérente si

1. $\exists x \in D_{X_i}\ f_{X_i}(x_i) = 0$ et $\forall X_j \in \texttt{voisins}(X_i)$, $\exists x_j \in D_{X_j}$ tel que $(f_{X_iX_j} \oplus f_{X_j})[X_i](x_i, x_j) = 0$
2. X_i est nœud cohérente

Le problème $\mathcal{P}$ est existentiellement arc-cohérent si toutes ses variables le sont.

Exemple 3.36 *Nous considérons le* WCSP $\mathcal{P}$ *sur trois variables* A, B, C *de domaine* $\{a, b\}$*. Avec les fonctions binaires* f_{AB}*,* f_{AC} *et* f_{BC} *et les fonctions unaires associées à chaque variable. Le* WCSP *est représenté à la figure 3.8a.*

La variable B *ne respecte pas la première condition de EAC. Il est donc possible de trouver une séquence de transformations permettant d'augmenter les coûts de la fonction unaire* f_B*. Pour cet exemple il suffit d'étendre le coût* $f_A(b)$ *sur* f_{AB} *et le coût* $f_C(b)$ *sur* f_{BC} *dont le résultat est donné par la figure 3.8b. Puis il est possible de projeter un coût de* 1 *sur* $f_B(a)$ *et sur* $f_B(b)$ *(*cf. *figure 3.8c). Il ne reste plus qu'à projeter la fonction unaire* f_B *sur* $f_\emptyset$ *pour que la variable* B *devienne nœud cohérente et que les deux conditions soient bien vérifiées pour toutes les variables du* WCSP*. Le* WCSP *présenté à la figure 3.8d est existentiellement arc-cohérent.*

Il est également possible de définir la cohérence d'arc existentielle et directionnelle (EDAC *Existential and Directional Arc Consistency*) car en pratique EAC est établie conjointement avec FDAC. La complexité de EDAC est de $O(md^2 \max(nd, \top))$ en temps et $O(md)$ en espace.

Définition 3.37 Cohérence d'arc existentielle et directionnelle
Une variable X est existentiellement et directionnellement arc-cohérente si

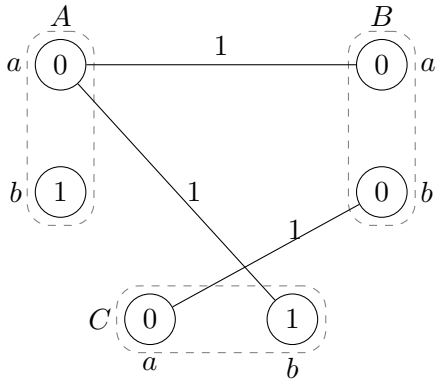
1. X est arc-cohérente
2. X est existentiellement et directionnellement arc-cohérente

Le problème $\mathcal{P}$ est *existentiellement et directionnellement arc-cohérent* si toutes ses variables le sont.

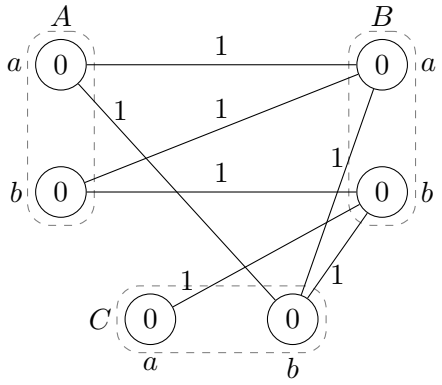
DAC et EDAC ont pu être étendues au cas où nous aurions des fonctions ternaires [Sanchez et al., 2008]. D'autres propositions existent pour des fonctions globales particulières [Lee and Leung, 2009] cependant leur complexité augmente de manière exponentielle en fonction de l'arité dans le cas général.

3.2.3.7 La cohérence d'arc optimale

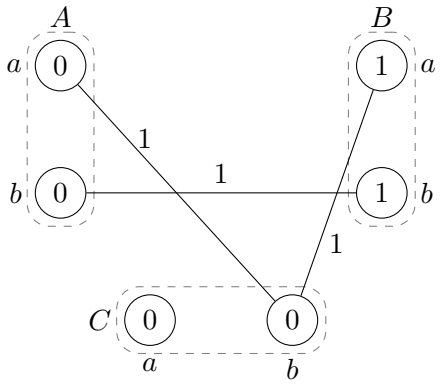
La cohérence d'arc optimale (OSAC, *Optimal Soft Arc Cohérence*) permet d'obtenir une reformulation équivalente dont $f_\emptyset$ est maximale. Elle abandonne l'obligation de conserver des coûts entiers pour s'autoriser des coûts fractionnaires. [Cooper et al., 2007] propose une formulation en système linéaire dont la solution est un ensemble de transformations simultanées permettant d'obtenir $f_\emptyset$ maximale.



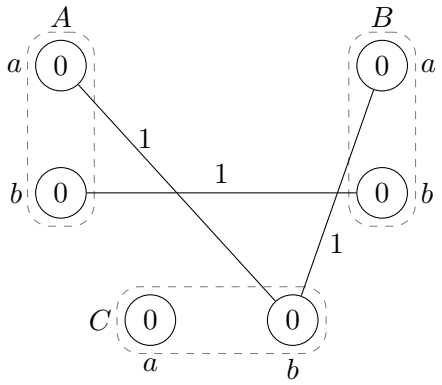
(a) Le problème $\mathcal{P}$ d'origine



(b) Transformation par les extensions des coûts $f_A(b)$ sur f_{AB} et $f_C(b)$ sur f_{BC}



(c) Transformation par les projections sur $f_B(a)$ et $f_B(b)$



(d) Transformation par la projection de f_B sur $f_\emptyset$

Figure 3.8 – Établissement de la cohérence d'arc existentielle

Cependant la résolution d'un tel système linéaire de grande taille est trop coûteuse pour être employée à chaque nœud de la recherche.

3.2.3.8 La cohérence d'arc virtuelle

La dernière cohérence locale en date est la cohérence d'arc virtuelle (VAC, *Virtual Arc Consistency*) [Cooper et al., 2008]. En passant par une transformation du WCSP en CSP cette cohérence trouve et applique une séquence de transformations permettant d'augmenter $f_\emptyset$, si elle existe, mais pas forcément de manière maximale. Cette cohérence locale a permis de résoudre deux instances d'un problème d'affectation de fréquences qui étaient non résolues depuis plus de dix ans [Cabon et al., 1999].

3.2.4 Autres simplifications

Il existe d'autres simplifications que les cohérences locales pour réduire la taille du problème.

3.2.4.1 Elimination de variables de degré borné

L'élimination de variables de degré borné `VE`(i) consiste à éliminer les variables dont leur degré (nombre de voisins) courant [3] est inférieur à une valeur i.

L'idée de [Larrosa, 2000] est d'introduire dans un algorithme DFBB une élimination de variables ayant un petit degré à chaque nœud de la recherche (élimination à la volée) : DFBB-`VE`(i). L'utilisation d'un degré de 2 maximum donne l'algorithme DFBB-`VE`(2) qui ne peut créer que des fonctions de coûts binaires qui sont bien propagées par les cohérences locales vues précédemment. De plus l'ordre dans lequel sont réalisées les éliminations de degré 2 n'a pas d'importance [Dechter, 2003].

3.2.4.2 Fusion de variables

Les fonctions de coûts binaires f_{XY} qui sont bijectives ou fonctionnelles (c'est-à-dire pour chaque variable X il n'existe qu'une seule valeur de Y compatible) permettent de substituer la variable X par la variable Y dans toutes les fonctions portant sur X.

Les contraintes modélisant des égalités entre variables sont des fonctions bijectives. Elles amènent à simplifier le problème en substituant les variables [Zhang et al., 2008].

3.2.5 Implémentation dans ToulBar2

ToulBar2 est un solveur de Wcsp en domaine public disponible sur la Forge logiciel de l'Inra Toulouse à l'adresse `http://mulcyber.toulouse.inra.fr/projects/toulbar2/`. Il intègre entre autres, l'ensemble des cohérences locales présentées. `VE`(2) à chaque nœud de la recherche est considéré dans la suite de cette thèse (voir le chapitre 4) comme étant une option incluse de l'algorithme DFBB, ainsi DFBB dénote en réalité DFBB-`VE`(2). Dans ToulBar2 il est également possible d'ajouter en pré-traitement de DFBB, de l'élimination de variable de degré i borné que nous noterons alors DFBB-`VE`(i). Il existe des méthodes de recherche utilisant également les décompositions arborescentes. Il peut également être utilisé pour la résolution des problèmes MPE pour les réseaux bayésiens en les transformant en Wcsp équivalents dont la technique de transformation est présentée dans la section suivante. Tous les développements issus des parties II et III sont inlcus dans la dernière version disponible de ToulBar2 (0.9.4.2).

3. il peut être modifié si d'autres variables ont été éliminées avant.

3.3 Lien avec les modèles graphiques probabilistes

Un $\mathbb{R}$-Wcsp est un Wcsp défini sur des réels positifs, les coûts peuvent ainsi représenter des fonctions réelles positives.

Définition 3.38 $\mathbb{R}$-Wcsp
Un $\mathbb{R}$-Wcsp est un réseau de fonctions de coûts associé à la structure de valuation $(\mathbb{R}^+, \oplus, \leqslant, 0, +\infty)$ où $\oplus$ est l'addition classique dans $\mathbb{R}$.

Un Pgm $(\mathbf{X}, \mathbf{D}, \mathbf{F})$ peut être traduit en un $\mathbb{R}$-Wcsp $(\mathbf{X}, \mathbf{D}, \mathbf{W})$ avec

$$\forall i \in [1, m],\ w_{\mathbf{S}_i} = -\log(f_{\mathbf{S}_i}) + C$$

où $w_{\mathbf{S}_i} \in \mathbf{W}$ et C est une constante pour garantir la positivité des fonctions de coûts obtenues. Dans le cas particulier des réseaux bayésiens, le lien avec les $\mathbb{R}$-Wcsp peut être formalisé avec la définition suivante.

Définition 3.39 Lien entre réseau bayésien et $\mathbb{R}$-Wcsp
Un réseau bayésien $\mathcal{B}$ défini sur un ensemble de variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ associé à la loi de probabilité $\mathbb{P}$ et au graphe orienté G peut se réécrire en un $\mathbb{R}$-Wcsp $(\mathbf{X}, \mathbf{D}, \mathbf{W})$ tel que

$$\mathbf{W} = \left\{ -\log\left(\mathbb{P}(X_1 \mid \texttt{parents}(X_1))\right), \ldots, -\log\left(\mathbb{P}(X_n \mid \texttt{parents}(X_n))\right) \right\}$$

et

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) = e^{-cout(\mathbf{x})}$$

Cependant en informatique, les nombres réels sont représentés sur un nombre fini de bit, et cela ne permet pas de représenter leur valeur exacte. Ainsi une succession de calculs sur ces nombres conduit à des résultats approchés dont nous ne maîtrisons pas l'écart à la valeur exacte [4]. Par contre la représentation des nombres entiers permet des calculs exacts.

Pour maîtriser cette erreur nous proposons d'approximer le Pgm par un Wcsp défini avec l'ensemble des fonctions suivantes :

$$\forall i \in [1, m],\ w_{\mathbf{S}_i} = \left\lceil -M \log(f_{\mathbf{S}_i}) + C \right\rceil$$

avec M et C deux constantes pour la précision et la positivité des $w_{\mathbf{S}_i}$ lors de la transformation des réels en entiers.

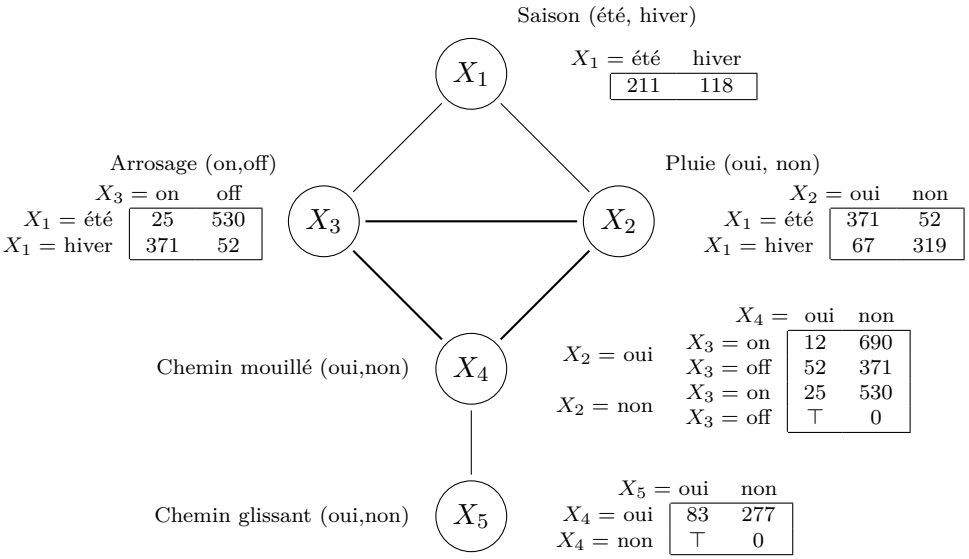
Pour les cas particuliers des réseaux bayésiens $C = 0$ car les fonctions sont des probabilités p variant dans l'intervalle [0,1] et nous garantit la positivité de $-\log(p)$. Nous pouvons écrire l'ensemble des fonctions de coûts $\mathbf{W}$ de la manière suivante :

4. Il existe tout de même des bibliothèques arithmétiques exactes sur les flottants

$$\mathbf{W} = \left\{ \left\lceil -M \log\left(\mathbb{P}(X_1 \mid \texttt{parents}(X_1))\right) \right\rceil, \ldots, \left\lceil -M \log\left(\mathbb{P}(X_n \mid \texttt{parents}(X_n))\right) \right\rceil \right\}$$

où $M = \dfrac{\log(10)}{\log(1-10^{-p})}$ et p correspond à la décimale à partir de laquelle le réel est tronqué.

Exemple 3.40 *Reprenons le réseau bayésien de la figure 2.1 page 23. En prenant $p = 2$ nous avons $M = -230$.*
Par exemple, pour la probabilité $\mathbb{P}(X_1 = \text{été}) = 0.4$ nous avons le coût $f_{X_1}(\text{été}) = \lceil -230 \cdot \log(0.4) \rceil = 211$. Ci-dessous le graphe de contraintes et les fonctions de coûts associées au Wcsp *traduisant le réseau bayésien. (En gras nous avons une fonction ternaire.)*



3.4 Synthèse

Dans ce chapitre nous avons présenté le formalisme des Wcsp qui étend celui des Csp pour représenter des fonctions de coûts dont nous voulons minimiser le coût pour ses solutions.

Nous avons présenté l'algorithme Dfbb qui est à la base de la résolution des Wcsp. Mais cette méthode nécessite de bons minorants qui peuvent être obtenus par des techniques de filtrage dites de cohérences locales. Cependant nous avons vu que ces cohérences locales ne peuvent être utilisées que sur des fonctions de petites arités (unaire, binaire, voire ternaire).

Le chapitre suivant propose de réduire l'arité des fonctions pour permettre une utilisation plus importante de ces cohérences qui ont démontré leur intérêt lors de leurs différentes évaluations au cours des années.

Nous avons décrit comment un réseau bayésien pouvait être transposé en WCSP, ce qui nous permettra dans la partie III d'utiliser toutes les techniques des WCSP à la résolution d'un problème biologique naturellement formalisé sous forme d'un réseau bayésien.

Partie II

Décompositions fonctionnelles et structurelles, exactes et approchées

Décomposition fonctionnelle : une décomposition par paire 4

Dans ce chapitre nous allons définir une nouvelle décomposition de fonctions pour les réseaux de fonctions de coûts permettant d'écrire une fonction comme étant une somme de fonctions de plus petites arités.

4.1 Etat de l'art

La factorisation de la contrainte globale `AllDiff` est la plus connue, elle consiste à découper la fonction en une clique de contraintes binaires de différence [1]. Par exemple, `AllDiff`(A,B,C,D) peut se réécrire avec l'ensemble des contraintes de différence suivantes $A \neq B$, $A \neq C$, $A \neq D$, $B \neq C$, $B \neq D$ et $C \neq D$.

La contrainte globale `AllEqual`, par sa structure mathématique (la transitivité de la relation d'égalité) va se décomposer en un arbre de contraintes binaires [2]. La contrainte `AllEqual`(A, B, C, D) peut se réécrire par exemple avec l'ensemble des contraintes d'égalités suivantes $A = B$, $B = C$ et $C = D$.

Des travaux existants ont considéré la factorisation de fonctions de probabilités spécifiques comme les *noisy-or* et ses généralisations [Heckerman and Breese, 1996]. Nous considérons une variable aléatoire binaire (un effet) E et $C_1, \ldots, C_n$ ses n parents, binaires également, dans un réseau bayésien (les causes de l'effet). La fonction de probabilités associée serait définie en théorie à partir de 2^n termes. Les fonctions de probabilités *noisy-or* ont la particularité que chaque cause C_i, indépendamment des autres, peut provoquer l'effet E (indépendance causale). La probabilité conditionnelle peut alors

1. Cependant établir GAC directement sur `AllDiff` n'est pas équivalent à établir AC sur cette décomposition. [Bessiere et al., 2009] prouvent qu'il n'existe pas de décomposition polynomiale même avec ajout de variables pour `AllDiff` garantissant cette équivalence de filtrage.
2. Pour cette contrainte nous avons vu dans le chapitre 3 que nous pouvions fusionner les variables car elles sont associées à des contraintes binaires bijectives.

s'écrire de la manière suivante

$$\mathbb{P}(E = faux \mid C_1 = c_1, \ldots, C_n = c_n) = (1 - \lambda) \prod_{c_i = true} \phi_i \tag{4.1}$$

Une variable de "bruit" est ajoutée, elle représente toutes les autres causes de X qui ne seraient pas représentées, sa probabilité d'être vraie est notée λ. Et chaque ϕ_i représente la probabilité d'échec de la cause C_i seule : $\phi_i = \mathbb{P}(E = faux \mid C_i = vrai, \forall j \neq i \; C_j = faux)$ Pour cette fonction, seules $n + 1$ probabilités sont nécessaires pour la définir complètement, ainsi $\mathbb{P}(E = vrai \mid C_1, \ldots, C_n) = 1 - \mathbb{P}(E = faux \mid C_1 = c_1, \ldots, C_n = c_n)$. Ainsi pour un effet ayant 10 causes possibles, 1 024 valeurs de probabilités auraient été nécessaires alors qu'avec cette structure *noisy-or* seules 11 valeurs de probabilités permettent de les calculer.

Exemple 4.1 *Considérons un diagnostic médical (extrait de [Darwiche, 2009]). Un patient a un mal de gorge, les causes possibles sont "attraper froid", "avoir la grippe" et "avoir une angine". En supposant que "attraper froid" (notée F), "avoir la grippe" (G) et "avoir une angine" (A) sont des causes indépendantes pour le "mal de gorge" (M) d'un patient, la table de la probabilité conditionnelle* $\mathbb{P}$(Mal de gorge | Attraper Froid, Avoir la Grippe, Avoir une Angine) *peut être déterminée à partir des quatre valeurs suivantes :*
$\mathbb{P}(M = faux \mid F = vrai, G = faux, A = faux) = 0.15$
$\mathbb{P}(M = faux \mid F = faux, G = vrai, A = faux) = 0.01$
$\mathbb{P}(M = faux \mid F = faux, G = faux, A = vrai) = 0.05$
$\lambda = 0.02$

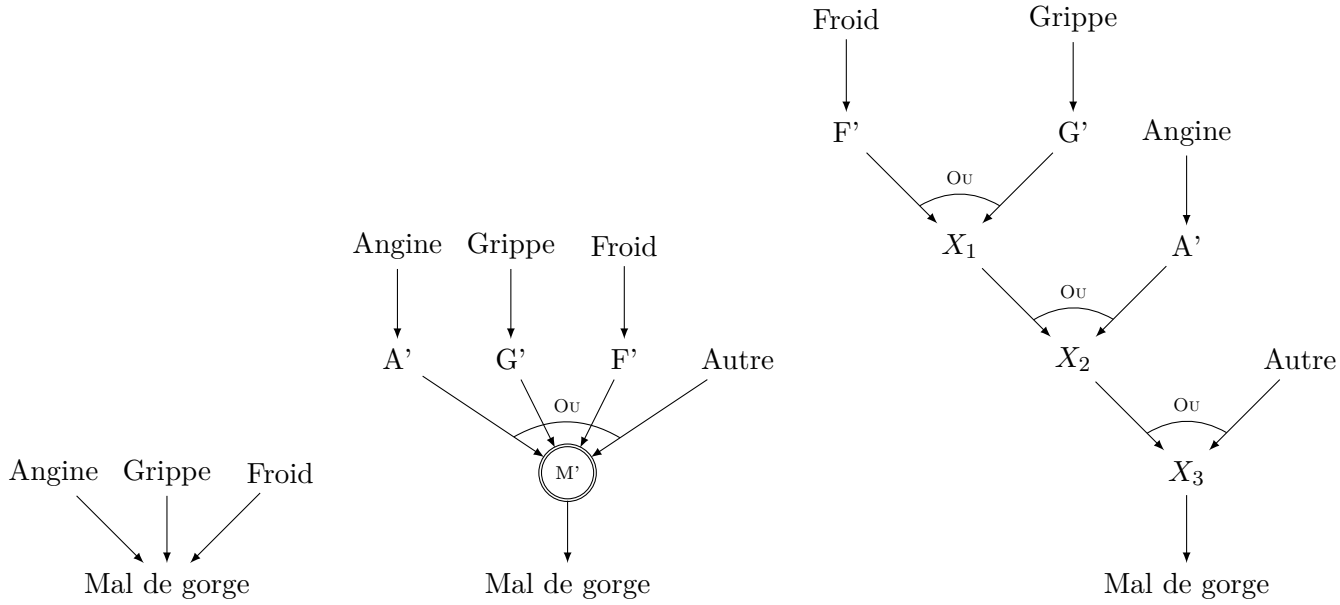
Par exemple,
$\mathbb{P}(M = vrai \mid F = vrai, G = vrai, A = vrai) = 1 - \big[(1 - 0.02)(0.15)(0.01)(0.05)\big] \approx 0.99993$

$\mathbb{P}(M = vrai \mid F = vrai, G = faux, A = vrai) = 1 - \big[(1 - 0.02)(0.15)(0.05)\big] \approx 0.993$

$\mathbb{P}(M = vrai \mid F = faux, G = faux, A = faux) = 1 - (1 - 0.02) = 0.02$

Cette fonction *noisy-or* peut se décomposer en ajoutant des variables intermédiaires. Ces ajouts de variables (*cf.* figure 4.1b) permettent d'obtenir une fonction déterministe représentant le Ou logique, cette structure en Ou induit également d'autres décompositions comme celle représentée à la figure 4.1c pour l'exemple précédent, les fonctions résultantes ne font intervenir que trois variables au maximum. De plus lorsque la variable effet E est affectée à $faux$ par l'équation (4.1) nous avons une décomposition de la probabilité en produit de facteurs. L'affectation d'une variable rendant d'autres indépendantes est appelée l'*indépendance contextuelle* (dépendant du contexte, de l'affectation de variables).

Zhang et Poole exploitent également les indépendances causales pour réduire les calculs nécessaires à l'algorithme d'élimination de variables pour l'inférence [Zhang and Poole, 1996]. Pour cela ils du-



(a) Réseau bayésien original **(b)** Apparition de la fonction déterministe OU **(c)** Décomposition d'un OU logique

Figure 4.1 – Différents réseaux de l'exemple 4.1

pliquent les variables "effet" et construisent des fonctions composées de l'effet et d'une de ses causes. Ainsi l'élimination d'une cause ne la relie pas à toutes les autres.

Savicky et Vomlel définissent des factorisations générales dédiées à l'inférence probabiliste [Savicky and Vomlel, 2007]. Ces factorisations consistent à décomposer une fonction en une somme de fonctions, chacune étant le produit d'un ensemble de fonctions unaires via l'ajout de variables intermédiaires dont leur domaine est au pire le produit cartésien de leurs parents. Les auteurs utilisent le calcul tensoriel pour obtenir ces factorisations.

Hammersley et Clifford proposent un théorème dans [Hammersley and Clifford, 1971] [3] disant qu'une distribution positive [4] $\mathbb{P}$ se factorise en un unique réseau de Markov minimal $\mathcal{H}$. Le réseau est minimal dans le sens où la suppression d'une arête crée une nouvelle indépendance conditionnelle non présente dans la loi $\mathbb{P}$. En prenant les cliques maximales dans $\mathcal{H}$, chaque clique donne la portée d'un facteur (une fonction positive) et $\mathbb{P}$ est égale au produit de ces facteurs divisé par une constante de normalisation. La décomposition des cliques n'est pas traitée, en effet certaines cliques pourraient être en réalité la réunion de fonctions plus petites (par exemple une clique de taille trois peut résulter de trois fonctions binaires portant sur trois variables ou une fonction ternaire portant sur ces trois mêmes variables).

Dans le cadre des CSP, si une contrainte de variables booléennes admet une décomposition en un

3. Voir aussi le Théorème 4.5 page 121 dans [Koller and Friedman, 2009].
4. aucune probabilité nulle

arbre de contraintes binaires, Dechter et Pearl proposent une méthode pour identifier cet arbre (il n'est pas forcément unique) en utilisant les arbres recouvrants de poids maximum [Dechter and Pearl, 1997] [5]. A partir d'une contrainte d'arité r, chaque poids $p(X_i, X_j)$ de l'arête $\{X_i, X_j\}$ est calculé suivant la formule

$$p(X_i, X_j) = \frac{1}{r} \sum_{\substack{x_i \in D_{X_i} \\ x_j \in D_{X_j}}} \#(x_i, x_j) \log \frac{\#(x_i, x_j)}{\#(x_i)\#(x_j)} \tag{4.2}$$

où $\#(x_i, x_j)$ est le nombre de tuples t autorisés par la contrainte tel que $t[X_i, X_j] = (x_i, x_j)$ même principe pour $\#(x_i)$.

Exemple 4.2 *Soit une contrainte sur les cinq variables A, B, C, D et E dont les tuples autorisés sont représentés dans la table suivante*

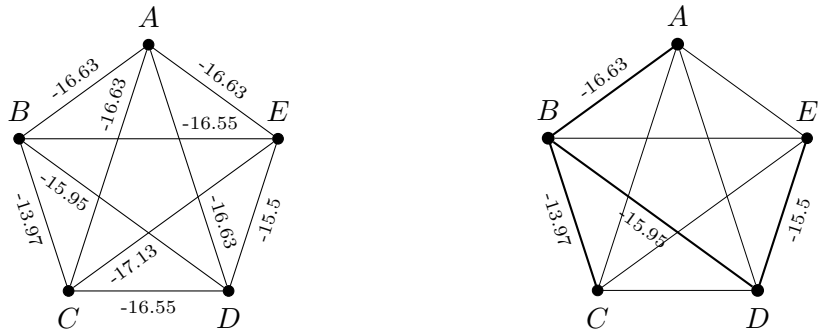
A	B	C	D	E
0	0	1	1	0
0	0	1	1	1
0	1	1	1	0
0	1	1	1	1
0	1	1	0	1
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1

A partir de la table nous pouvons calculer par exemple $\#(A = 0) = 8$, $\#(B = 1) = 6$ ou encore $\#(B = 0, C = 1) = 2$, $\#(D = 1, E = 1) = 3$. Sur la figure 4.2a sont représentés les poids qui sont calculés à partir de la formule (4.2). La figure 4.2b représente un arbre recouvrant de poids maximum identifiant ainsi les contraintes binaires le composant C_{AB}, C_{BC}, C_{BD} et C_{DE}. Les tuples autorisés pour chacune des contraintes sont la projection de la contrainte d'origine sur la portée des contraintes considérées.

Cette décomposition permet de réduire le nombre d'arêtes du graphe de contrainte et également d'en réduire sa largeur d'arbre. De plus, faire GAC sur la contrainte ou AC sur sa décomposition est équivalent. La contrainte `AllEqual` que nous avons énoncé précédemment fait partie de ces contraintes décomposables en arbre de contraintes binaires.

Il existe d'autres moyens que la décomposition pour réduire l'arité des fonctions. Par exemple, il est toujours possible de transformer un Wcsp n-aire en un Wcsp binaire par la transformation duale. Chaque fonction de coûts du Wcsp devient une variable dans le dual ayant comme domaine l'ensemble

5. Lorsque la contrainte n'admet pas de décompsition en arbre, les arbres qui peuvent être construits ne résument pas la contrainte.



(a) Graphe de la contrainte

(b) Un arbre recouvrant de poids maximum

Figure 4.2 – Construction d'un arbre de contraintes

des tuples. Pour ce qui est des fonctions de coûts dans le dual, nous avons une fonction unaire pour chaque variable dans le dual où $f_{C_i}(c_i)$ correspond au coût associé au tuple c_i dans la fonction C_i du Wcsp d'origine et il existe une contrainte entre deux variables C_i et C_j dans le dual lorsque les deux fonctions de coût partagent au moins une variable dans le Wcsp d'origine. Le tuple (c_i, c_j) est autorisé si l'ensemble des variables (d'origine) communes à C_i et C_j ont la même valeur dans c_i et c_j.

Cependant avec cette nouvelle représentation nous obtenons des variables ayant des domaines de grande taille (d^r, si d est le domaine maximum des variables d'origine et r l'arité maximale de leurs fonctions). Cette reformulation a été testée dans [de Givry et al., 2003] pour Max-3Sat, elle ne fait rien gagner en terme de complexité de la propagation et était apparemment inefficace.

La décomposition par paire que nous proposons dans le cadre Wcsp permet d'enlever (au moins) une arête à chaque clique du graphe issue de la fonction de coûts décomposable, mais elle ne garantit pas que nous ayons un filtrage par cohérence locale équivalent à celui de la fonction de départ.

4.2 Décomposition par paire d'une fonction de coûts

Nous définissons d'abord la notion de *décomposition par paire*.

Définition 4.3 Une *décomposition par paire* d'une fonction de coûts $f(\mathbf{S})$ par rapport à deux variables $X, Y \in \mathbf{S}$ ($X \neq Y$), est une réécriture de f en une somme de deux fonctions (positives) $f_1(\mathbf{S} \setminus \{Y\})$ et $f_2(\mathbf{S} \setminus \{X\})$ telles que :

$$f(\mathbf{S}) = f_1(\mathbf{S} \setminus \{Y\}) + f_2(\mathbf{S} \setminus \{X\})$$

L'exemple 4.4 page suivante présente une fonction décomposable. Une décomposition par paire remplace une fonction de coûts d'arité $r = |\mathbf{S}|$ par deux fonctions de coûts d'arité $(r-1)$. Ce processus

de décomposition peut être répété récursivement sur les fonctions résultantes, jusqu'à ce qu'aucune décomposition par paire ne soit trouvée pour chaque fonction de coûts restante.

Exemple 4.4 *La fonction $f(X,Y)$ suivante est décomposable par paire par rapport à X et Y.*

X	Y	$f(X,Y)$
a	a	5
a	b	7
b	a	4
b	b	6

$=$

X	$f_1(X)$
a	2
b	1

$+$

Y	$f_2(Y)$
a	3
b	5

Définition 4.5 Une fonction de coûts $f(\mathbf{S})$ qui ne peut être décomposée par paire pour aucun choix de $X, Y \in \mathbf{S}$ est dite *irréductible*.

Propriété 4.6 *Un* WCSP *est* décomposé par paire *si toutes ses fonctions de coûts sont irréductibles*

Cette propriété est respectée en appliquant le processus itératif précédemment décrit pour toutes les fonctions de coûts.

Le théorème suivant montre que tester si une fonction de coûts peut être décomposée par paire par rapport à X, Y est équivalent à tester l'indépendance de paire entre X et Y dans une distribution de probabilité appropriée.

Théorème 4.7 *Soit $\mathbf{S} = \{X,Y\} \cup \mathbf{Z}$ un ensemble de variables aléatoires, $f(\mathbf{S})$ une fonction de coûts sur $\mathbf{S}$ avec au moins une affectation autorisée, et une* distribution de Gibbs *paramétrisée par f $\mathbb{P}_f = \frac{1}{\sum_{\mathbf{S}} \exp(-f(\mathbf{S}))} \exp(-f(\mathbf{S}))$. Alors,*

$$(X \perp\!\!\!\perp Y \mid \mathbf{Z}) \iff f(X,\mathbf{Z},Y) = f_1(X,\mathbf{Z}) \oplus f_2(\mathbf{Z},Y)$$

Démonstration. Dans la suite, nous utilisons $\mathbb{P}$ comme raccourci pour $\mathbb{P}_f$ et $C = \sum_{\mathbf{S}} \exp(-f(\mathbf{S})) > 0$, une constante de normalisation. Par définition des indépendances conditionnelles [Lauritzen, 1996], nous avons :

$$\begin{aligned}(X \perp\!\!\!\perp Y \mid \mathbf{Z}) &\iff \mathbb{P}(X,Y,\mathbf{Z}) = \mathbb{P}(X \mid \mathbf{Z})\mathbb{P}(Y \mid \mathbf{Z})\mathbb{P}(\mathbf{Z}) \\ &\iff \mathbb{P}(X,Y,\mathbf{Z}) = \frac{\mathbb{P}(X,\mathbf{Z})}{\mathbb{P}(\mathbf{Z})}\mathbb{P}(Y,\mathbf{Z}), \text{ avec } \mathbb{P}(\mathbf{Z}) > 0\end{aligned}$$

$\boxed{\Longrightarrow}$

Supposons $\mathbb{P}(X,Y,\mathbf{Z}) = \mathbb{P}(X \mid \mathbf{Z})\ \mathbb{P}(Y \mid \mathbf{Z})\ \mathbb{P}(\mathbf{Z})$.

Nous avons $f(X,\mathbf{Z},Y) = -\log(C\ \mathbb{P}(X,Y,\mathbf{Z})) = -\log(C\ \mathbb{P}(X \mid \mathbf{Z})\ \mathbb{P}(Y \mid \mathbf{Z})\ \mathbb{P}(\mathbf{Z}))$.

Définissons $f_1'(X,\mathbf{Z}) = -\log(\mathbb{P}(X \mid \mathbf{Z}))$ et $f_2'(\mathbf{Z},Y) = -\log(\mathbb{P}(Y \mid \mathbf{Z})\mathbb{P}(\mathbf{Z}))$, deux fonctions de coût positives.

Ainsi $f(X,\mathbf{Z},Y) = f_1'(X,\mathbf{Z}) + f_2'(\mathbf{Z},Y) - \log(C)$. Puisque f est positive, nous avons $\forall x \in D_X, \forall y \in D_Y, \forall z \in D_{\mathbf{Z}}, f_1'(x,z) + f_2'(z,y) - \log(C) \geq 0$.

Si $\log(C)$ est négatif, nous ajoutons $-\log(C)$ à f_1' ou f_2' afin d'obtenir f_1, f_2.

Sinon, pour chaque $z \in D_{\mathbf{Z}}$, nous décomposons $\log(C) = c_z^1 + c_z^2$ en deux nombres positifs.

Soit $\hat{x}_z = \underset{x \in D_X}{\operatorname{argmin}} f_1'(x, z)$ et $\hat{y}_z = \operatorname{argmin}_{y \in D_Y} f_2'(z, y)$.

De plus, nous avons $f_1'(\hat{x}_z, z) + f_2'(z, \hat{y}_z) - \log(C) = f(\hat{x}_z, z, \hat{y}_z) \geq 0$.

Une solution possible est $c_z^1 = \min(f_1'(\hat{x}_z, z), \log(C))$ et $c_z^2 = \log(C) - c_z^1$.

Soit $c_z^1 = \log(C) \leq f_1'(\hat{x}_z, z)$ et $c_z^2 = 0$, soit $c_z^1 = f_1'(\hat{x}_z, z)$ et $c_z^2 = \log(C) - f_1'(\hat{x}_z, z)$.

Dans ce cas, $f_2'(z, \hat{y}_z) - c_z^2 = f_2'(z, \hat{y}_z) + f_1'(\hat{x}_z, z) - \log(C) \geq 0$, ainsi $\forall y \in D_Y, f_2'(z, y) - c_z^2 \geq 0$.

Nous définissons $\forall x, y, z,\ f_1(x, z) = f_1'(x, z) - c_z^1$ et $f_2(z, y) = f_2'(z, y) - c_z^2$, qui sont des nombres positifs.

Finalement, nous en déduisons $f(X, \mathbf{Z}, Y) = f_1(X, \mathbf{Z}) + f_2(\mathbf{Z}, Y)$.

$\boxed{\Longleftarrow}$

Supposons que nous ayons trois fonctions de coûts $f(X, \mathbf{Z}, Y), f_1(X, \mathbf{Z}), f_2(\mathbf{Z}, Y)$ telles que $f(X, \mathbf{Z}, Y) = f_1(X, \mathbf{Z}) + f_2(\mathbf{Z}, Y)$.

Nous avons
$\exp(-f(X, \mathbf{Z}, Y)) = \exp(-f_1(X, \mathbf{Z})) \exp(-f_2(\mathbf{Z}, Y))$.
Par marginalisation, nous avons

$$\begin{aligned}
\mathbb{P}(X, \mathbf{Z}) &= \sum_Y \mathbb{P}(X, Y, \mathbf{Z}) \\
&= \frac{1}{C} \exp(-f_1(X, \mathbf{Z})(\sum_Y \exp(-f_2(\mathbf{Z}, Y)))
\end{aligned}$$

$$\begin{aligned}
\mathbb{P}(Y, \mathbf{Z}) &= \sum_X \mathbb{P}(X, Y, \mathbf{Z}) \\
&= \frac{1}{C} \exp(-f_2(\mathbf{Z}, Y))\ (\sum_X \exp(-f_1(X, \mathbf{Z})))
\end{aligned}$$

$$\begin{aligned}
\text{et } \mathbb{P}(\mathbf{Z}) &= \sum_{X,Y} \mathbb{P}(X, Y, \mathbf{Z}) \\
&= \frac{1}{C} \sum_X \exp(-f_1(X, \mathbf{Z})))(\sum_Y \exp(-f_2(\mathbf{Z}, Y)))
\end{aligned}$$

Finalement, nous en déduisons $\dfrac{\mathbb{P}(X, \mathbf{Z})}{\mathbb{P}(\mathbf{Z})} \mathbb{P}(Y, \mathbf{Z}) = \frac{1}{C} \exp(-f_1(X, \mathbf{Z})) \exp(-f_2(\mathbf{Z}, Y)) = \mathbb{P}(X, Y, \mathbf{Z})$. □

Au lieu de tester l'indépendance de paire dans $\mathbb{P}_f$, qui nécessite des sommations et des multiplications des nombres réels, nous proposons un test plus simple pour la décomposabilité de paire basé sur des égalités de différences de coûts.

La section suivante détaille ce test pour les fonctions de coûts finis. La section d'après propose une nouvelle algèbre pour prendre en compte les coûts infinis.

4.2.1 Cas d'une fonction de coûts finis

Théorème 4.8 ***Décomposabilité d'une fonction***
Une fonction de coûts finis $f(X, \mathbf{Z}, Y)$ est décomposable par paire par rapport à X et Y si et seulement si $\forall u, v \in D_X$, $\forall \mathbf{z} \in D_{\mathbf{Z}}$ et $\forall k, l \in D_Y (k < l)$ [6] *:*

$$f(u, \mathbf{z}, k) - f(u, \mathbf{z}, l) = f(v, \mathbf{z}, k) - f(v, \mathbf{z}, l)$$

Démonstration. La fonction $f(X, \mathbf{Z}, Y)$ est à coûts finis donc l'addition utilisée est l'opérateur classique des entiers. Soient $D_X = \{1, 2, \ldots, d_X\}$ et $D_Y = \{1, 2, \ldots, d_Y\}$

$f(X, \mathbf{Z}, Y)$ est décomposable en $f_1(X, \mathbf{Z})$ et $f_2(\mathbf{Z}, Y)$ si et seulement si tous les systèmes $(\mathcal{S}_{\mathbf{z}})$ suivants $(\mathbf{z} \in \mathbf{Z})$ admettent une solution.

$$(\mathcal{S}_{\mathbf{z}}) \begin{cases} f_1(1, \mathbf{z}) + f_2(\mathbf{z}, 1) = f(1, \mathbf{z}, 1) \\ f_1(x, \mathbf{z}) + f_2(\mathbf{z}, y) = f(x, \mathbf{z}, y) \quad \forall x \in D_X, y \in D_Y \\ f_1(d_X, \mathbf{z}) + f_2(\mathbf{z}, d_Y) = f(d_X, \mathbf{z}, d_Y) \end{cases}$$

$(\mathcal{S}_{\mathbf{z}})$ peut se réécrire de manière équivalente :
$\forall u, v \in D_X$ et $\forall k, l \in D_Y$

$$(\mathcal{R}_{\mathbf{z}}) \begin{cases} f_1(u, \mathbf{z}) + f_2(\mathbf{z}, k) = f(u, \mathbf{z}, k) \\ f_1(v, \mathbf{z}) + f_2(\mathbf{z}, k) = f(v, \mathbf{z}, k) \\ f_1(u, \mathbf{z}) + f_2(\mathbf{z}, l) = f(u, \mathbf{z}, l) \\ f_1(v, \mathbf{z}) + f_2(\mathbf{z}, l) = f(v, \mathbf{z}, l) \end{cases}$$

$$(\mathcal{R}_{\mathbf{z}}) \Longleftrightarrow \begin{cases} f_2(\mathbf{z}, k) = f(u, \mathbf{z}, k) - f_1(u, \mathbf{z}) \\ f_2(\mathbf{z}, k) = f(v, \mathbf{z}, k) - f_1(v, \mathbf{z}) \\ f_1(u, \mathbf{z}) = f(u, \mathbf{z}, l) - f_2(\mathbf{z}, l) \\ f_1(v, \mathbf{z}) = f(v, \mathbf{z}, l) - f_2(\mathbf{z}, l) \end{cases}$$

$$(\mathcal{R}_{\mathbf{z}}) \Longleftrightarrow \begin{cases} f_1(u, \mathbf{z}) = f(u, \mathbf{z}, l) - f_2(\mathbf{z}, l) \\ f_1(v, \mathbf{z}) = f(v, \mathbf{z}, l) - f_2(\mathbf{z}, l) \\ f_2(\mathbf{z}, k) = f(u, \mathbf{z}, k) - f(u, \mathbf{z}, l) + f_2(\mathbf{z}, l) \quad (\star) \\ f_2(\mathbf{z}, k) = f(v, \mathbf{z}, k) - f(v, \mathbf{z}, l) + f_2(\mathbf{z}, l) \quad (*) \end{cases}$$

Avec les lignes $(\star)$ et $(*)$ nous pouvons en déduire que $(\mathcal{R}_{\mathbf{z}})$ admet une solution si est seulement si

$$f(u, \mathbf{z}, k) - f(u, \mathbf{z}, l) = f(v, \mathbf{z}, k) - f(v, \mathbf{z}, l)$$

□

Exemple 4.9 *Soit la fonction $f(A, B, C)$ avec la table de coûts suivante. Testons sa décomposabilité par rapport à A et C.*

6. Nous avons $-(f(u, z, k) - f(u, z, l)) = f(u, z, l) - f(u, z, k)$ d'où la condition de tester que dans un sens les égalités en prenant un ordre arbitraire sur les valeurs.

A	B	C	f
0	0	0	1
0	0	1	0
0	1	0	4
0	1	1	1
1	0	0	3
1	0	1	2
1	1	0	7
1	1	1	4

Il faut tester les égalités suivantes :

$f(0,0,0) - f(0,0,1) = f(1,0,0) - f(1,0,1) \mid 1-0 = 3-2$ ✓

$f(0,1,0) - f(0,1,1) = f(1,1,0) - f(1,1,1) \mid 4-1 = 7-4$ ✓

Les égalités sont vérifiées la fonction est donc décomposable par rapport à A et C.

A	B	$f_1(A,B)$
0	0	0
0	1	1
1	0	2
1	1	4

B	C	$f_2(B,C)$
0	0	1
0	1	0
1	0	3
1	1	0

Ce théorème ne peut pas s'appliquer lorsque les fonctions de coûts admettent des coûts infinis. En effet dans l'exemple suivant, une égalité n'est pas vérifiée pourtant la fonction est décomposable.

Exemple 4.10 *Soit la fonction $f(A,B,C)$ avec la table de coût suivante. Testons sa décomposabilité par rapport à A et C.*

A	B	C	f
0	0	0	$\top$
0	0	1	$\top$
0	1	0	4
0	1	1	1
1	0	0	3
1	0	1	2
1	1	0	7
1	1	1	4

Il faut tester les égalités suivantes :

$f(0,0,0) - f(0,0,1) = f(1,0,0) - f(1,0,1) \mid \top - \top = 3-2$ ✗

$f(0,1,0) - f(0,1,1) = f(1,1,0) - f(1,1,1) \mid 4-1 = 7-4$ ✓

En effet $\top - \top = \top \neq 3-2 = 1$. Cependant la fonction est décomposable et voici deux fonctions f_1 et f_2 possibles telles que $f = f_1 \oplus f_2$.

A	B	f_1
0	0	$\top$
0	1	1
1	0	2
1	1	4

B	C	f_2
0	0	1
0	1	0
1	0	3
1	1	0

4.2.2 Intégration des coûts infinis (ou contraintes)

Pour obtenir un théorème de décomposabilité qui puisse exploiter les coûts infinis, nous introduisons une extension de la structure de valuation de coûts $E = \mathbb{Z} \cup \{-\top, \top, \Omega\}$ incluant les coûts négatifs et un élément spécial absorbant Ω qui capture la liberté offerte par des différences de coûts infinis lors du test de décomposabilité.

4.2.2.1 Algèbre dans E

Pour cette algèbre nous définissons deux lois de composition internes, l'addition $\boxplus$ et la soustraction $\boxminus$ ainsi qu'une nouvelle relation binaire pour comparer deux éléments de E.

L'addition

Définition 4.11

$$a \boxplus b = \begin{cases} a+b \quad si \quad -\top < a+b < \top \quad et \quad a,b \in \mathbb{Z} & (1) \\ \top \quad si \quad \begin{cases} a+b \geqslant \top \quad et \quad a,b \neq \top,-\top \\ a=\top \quad et \quad b \neq -\top \\ b=\top \quad et \quad a \neq -\top \end{cases} & (2) \\ -\top \quad si \quad \begin{cases} a+b \leqslant -\top \quad et \quad a,b \in \mathbb{Z} \\ a=-\top \quad et \quad b \neq \top \\ b=-\top \quad et \quad a \neq \top \end{cases} & (3) \\ \Omega \quad si \quad \begin{cases} a=\top \quad et \quad b=-\top \\ b=\top \quad et \quad a=-\top \\ a=\Omega \quad ou \quad b=\Omega \end{cases} & (4) \end{cases}$$

Propriété 4.12
La loi de composition interne $\boxplus$ est commutative, associative, admet comme élément neutre 0 et Ω comme élément absorbant.

Démonstration.

commutativité : $\forall a,b \in E,\ (a \boxplus b) = b \boxplus a$

1. $a+b = b+a$ par définition de $\mathbb{Z}$.
2. $a+b \geqslant \top \equiv b+a \geqslant \top$ par définition de $\mathbb{Z}$.
3. $a+b \leqslant -\top \equiv b+a \leqslant -\top$ par définition de $\mathbb{Z}$.
4. $T \boxplus -T = \Omega = -T \boxplus T$
   $\forall a \in E,\ a \boxplus \Omega = \Omega = \Omega \boxplus a$

$\Longrightarrow \boxplus$ est une loi commutative.

associativité $\forall a,b,c \in E,\ (a \boxplus b) \boxplus c = a \boxplus (b \boxplus c)$

1. $-\top < a+(b+c) < \top$ par définition de $\mathbb{Z}$: $a+(b+c) = (a+b)+c$

2. Si $a, b, c \in \mathbb{Z}$, $a + (b + c) \geqslant \top \equiv (a + b) + c \geqslant \top : a \boxplus (b \boxplus c) = \top = (a \boxplus b) \boxplus c)$
   Si $a = \top$ $b, c \in \mathbb{Z} \cup \top$, $\top \boxplus (b \boxplus c) = \top$ et $(\top \boxplus b) \boxplus c = \top \boxplus c = \top$
   Si $b = \top$ $a, c \in \mathbb{Z} \cup \top$, $a \boxplus (\top \boxplus c) = a \boxplus \top = \top$ et $(a \boxplus \top) \boxplus c = \top \boxplus c = \top$
   Si $c = \top$ $a, b \in \mathbb{Z} \cup \top$, $a \boxplus (b \boxplus \top) = a \boxplus \top = \top$ et $(a \boxplus b) \boxplus \top = \top$
3. Même principe que le point précédent.
4. $\top \boxplus (\top \boxplus -\top) = \top \boxplus \Omega = \Omega$ et $(\top \boxplus \top) \boxplus -\top = \top \boxplus -\top = \Omega$
   $\top \boxplus (-\top \boxplus \top) = \top \boxplus \Omega = \Omega$ et $(\top \boxplus -\top) \boxplus \top = \Omega \boxplus \top = \Omega$
   Si $a \in \mathbb{Z}$, $a \boxplus (\top \boxplus -\top) = a \boxplus \Omega = \Omega$ et $(a \boxplus \top) \boxplus -\top = \top \boxplus -\top = \Omega$
   Si $b \in \mathbb{Z}$, $\top \boxplus (b \boxplus -\top) = \top \boxplus -\top = \Omega$ et $(\top \boxplus b) \boxplus -\top = \top \boxplus -\top = \Omega$
   Si $c \in \mathbb{Z}$,$\top \boxplus (-\top \boxplus c) = \top \boxplus -\top = \Omega$ et $(\top \boxplus -\top) \boxplus c = \Omega \boxplus c = \Omega$

   $\forall a, b \in E$,
   $a \boxplus (\Omega \boxplus b) = a \boxplus \Omega = \Omega$ et $(a \boxplus \Omega) \boxplus b = \Omega \boxplus b = \Omega$
   $a \boxplus (b \boxplus \Omega) = a \boxplus \Omega = \Omega$ et $(a \boxplus b) \boxplus \Omega = \Omega$
   $\Omega \boxplus (a \boxplus b) = \Omega$ et $(\Omega \boxplus a) \boxplus b = \Omega$

$\Longrightarrow \boxplus$ est une loi associative.

élément neutre $\forall a \in E$, $(a \boxplus 0) = a$

1. $a \boxplus 0 = a$ car 0 est l'élément neutre de $\mathbb{Z}$.
2. $\top \boxplus 0 = \top$ par définition de la loi.
3. $-\top \boxplus 0 = -\top$ par définition de la loi.
4. $\Omega \boxplus 0 = \Omega$ par définition de la loi.

$\Longrightarrow$ la loi $\boxplus$ admet l'élément neutre 0.

élément absorbant $\forall a \in E$, $(a \boxplus \Omega) = \Omega$ par définition de la loi $\forall a \in E$, $a \boxplus \Omega = \Omega$ donc Ω est absorbant.

□

La soustraction

Définition 4.13

$$a \boxminus b = \begin{cases} a - b & si \quad -T < a - b < \top \quad et \quad a, b \in \mathbb{Z} & (1) \\ \top & si \quad \begin{cases} a = \top & et \quad b \in \mathbb{Z} \cup -\top \\ b = -\top & et \quad a \in \mathbb{Z} \cup \top \end{cases} & (2) \\ -\top & si \quad \begin{cases} b = \top & et \quad a \in \mathbb{Z} \cup -\top \\ a = -\top & et \quad b \in \mathbb{Z} \cup \top \end{cases} & (3) \\ \Omega & si \quad \begin{cases} a = \top & et \quad b = \top \\ a = -\top & et \quad b = -\top \\ a = \Omega & ou \quad b = \Omega \end{cases} & (4) \end{cases}$$

Propriété 4.14
La loi de composition interne $\boxminus$ admet 0 comme élément neutre, Ω comme élément absorbant à gauche et à droite et chaque élément de E admet un opposé.

Démonstration. **élément neutre :** $\forall a \in E, (a \boxminus 0) = a$

1. $a \boxminus 0 = a$ car 0 est l'élément neutre de $\mathbb{Z}$.
2. $\top \boxminus 0 = \top$ par définition de la loi.
3. $(-\top) \boxminus 0 = -\top$ par définition de la loi.
4. $\Omega \boxminus 0 = \Omega$ par définition de la loi.

opposé : $\forall a \in E \setminus \Omega, (0 \boxminus a) = -a$

1. $0 \boxminus a = 0 - a = -a$ par la définition de $\mathbb{Z}$.
2. $0 \boxminus \top = -\top$ par définition de la loi.
3. $0 \boxminus (-\top) = \top$ par définition de la loi.
4. $0 \boxminus \Omega = \Omega$

élément absorbant à droite : $\forall a \in E, (a \boxminus \Omega) = \Omega$ (*cf.* (4) de la définition 4.13).

élément absorbant à gauche : $\forall a \in E, (\Omega \boxminus a) = \Omega$ (*cf.* (4) de la définition 4.13).

□

Propriété 4.15 $a \boxplus (-b) = a \boxminus b$

Démonstration.

1. Si $a = \Omega$ ou $b = \Omega$, $a \boxplus (-b) = \Omega$ et $a \boxminus b = \Omega$
2. Si $a = \top$ (resp. $-\top$), $b \in \mathbb{Z}$, $a \boxplus (-b) = \top$ (resp. $-\top$) et $a \boxminus b = \top$ (resp. $-\top$)
3. Si $a \in \mathbb{Z}$, $b = \top$ (resp. $-\top$), $a \boxplus (-b) = -\top$ (resp. $\top$) et $a \boxminus b = -\top$ (resp. $\top$)
4. Si $a = \top$ et $b = -\top$, $\top \boxplus (-(-\top)) = \top \boxplus (\top) = \top$ et $\top \boxminus (-\top) = \top$

dans tous les cas $a \boxplus (-b) = a \boxminus b$ □

L'égalité

Définition 4.16 La relation binaire $\mathring{=}$ sur E est définie lorsqu'une des conditions suivantes est vérifiée :

1. $a, b \in \mathbb{Z}$ et $a = b$
2. $a = \top$ et $b = \top$
3. $a = -\top$ et $b = -\top$
4. $a = \Omega$ ou $b = \Omega$

Remarque 4.17 *La relation binaire* $\stackrel{\circ}{=}$ *n'est pas transitive, en effet* $2 \stackrel{\circ}{=} \Omega$ *et* $\Omega \stackrel{\circ}{=} 3$ *mais* $2 \not\stackrel{\circ}{=} 3$.

Pour terminer cette algébre, les propriétés suivantes font référence à l'ensemble de définition des fonctions de coûts.

Propriété 4.18 *Soit* $E^+ = \mathbb{N} \cup \{\top\}$, $\forall a, b \in E^+$,

1. $(a \boxplus b) \equiv (a \oplus b)$
2. $(a = b) \equiv (a \stackrel{\circ}{=} b)$
3. $(a \boxminus b = \Omega) \not\equiv (a \boxminus b \stackrel{\circ}{=} \Omega)$

Démonstration.

1. – $\forall a, b \in \mathbb{N}$,
   $a + b < \top$: $a \boxplus b = a + b$ et $a \oplus b = a + b$
   $a + b \geqslant \top$: $a \boxplus b = \top$ et $a \oplus b = \top$
   – $\forall b \in E^+$ $\top \boxplus b = \top$ et $\top \oplus b = \top$
2. $a, b \in \mathbb{N}$, par définition $(a = b) \equiv (a \stackrel{\circ}{=} b)$.
   $a = \top$ et $b = \top$ donc $a = b$ et $a \stackrel{\circ}{=} b$.
3. D'aprés la définition de $\boxminus$: $a \boxminus b = \Omega$ implique $a = b = \top$, mais $a \boxplus b \stackrel{\circ}{=} \Omega$ est toujours vraie pour tout a, b.
   Par exemple, $\top \boxminus 5 \neq \Omega$ mais $\top \boxminus 5 \stackrel{\circ}{=} \Omega$ (par définition)

□

4.2.2.2 Décomposabilité

Il est toujours possible de décomposer une fonction de coûts f en trois fonctions de coûts (positives) $f(X, \mathbf{Z}, Y) = f_1(X, \mathbf{Z}) + f_2(\mathbf{Z}, Y) + \Delta(X, \mathbf{Z}, Y)$ (f_1, f_2 peuvent être égales à la fonction constante nulle). Une fonction de coût non nulle est appelée *réductible* si f_1 ou f_2 sont des fonctions de coûts non nulles. Sinon elle est appelée *irréductible*. Nous avons une condition spécifique pour tester la *réductibilité*.

Propriété 4.19 *Une fonction de coût non nulle* $f(X, \mathbf{Z}, Y)$ *est* réductible *si et seulement si*

$\exists x \in D_X, \exists z \in D_{\mathbf{Z}}$ *t.q.* $\min_{y \in D_Y} f(x, z, y) > 0$ *(i.e.* $f_1 = f[X, \mathbf{Z}] \neq 0$*)*

ou

$\exists y \in D_Y, \exists z \in D_{\mathbf{Z}}$ *t.q.* $\min_{x \in D_X} f(x, z, y) > 0$ *(i.e.* $f_2 = f[\mathbf{Z}, Y] \neq 0$*).*

Démonstration. Soit $f(X, \mathbf{Z}, Y)$ une fonction réductible. Il existe $f_1(X, \mathbf{Z})$, $f_2(\mathbf{Z}, Y)$, $\Delta(X, \mathbf{Z}, Y)$ décomposant $f(X, \mathbf{Z}, Y)$ telles que $f_1(X, \mathbf{Z})$ ou $f_2(\mathbf{Z}, Y)$ ne soit pas la fonction nulle.

Supposons que $f_1(X, \mathbf{Z})$ ne soit pas la fonction nulle. Il existe alors i, j tels que $f_1(i, j) > 0$.
De plus nous avons $\forall k \in D_Y$, $f(i, j, k) = f_1(i, j) \oplus f_2(j, k) \oplus \Delta(i, j, k)$.

Or $\forall k \in D_Y,\ f_2(j,k) \geqslant 0$ et $\Delta(i,j,k) \geqslant 0$ donc $f(i,j,k) = f_1(i,j) \oplus f_2(j,k) \oplus \Delta(i,j,k) > 0$ d'où $\min_{k \in D_Y} f(i,j,k) > 0$.
Supposons que $f_2(\mathbf{Z}, Y)$ ne soit pas la fonction nulle. Il existe alors j,k tels que $f_2(j,k) > 0$.
De plus nous avons $\forall i \in D_X,\ f(i,j,k) = f_1(ij) \oplus f_2(jk) \oplus \Delta(i,j,k)$.
Or $\forall i \in D_X,\ f_1(i,j) \geqslant 0$ et $\Delta(i,j,k) \geqslant 0$ donc $f(i,j,k) = f_1(i,j) \oplus f_2(j,k) \oplus \Delta(i,j,k) > 0$ d'où $\min_{i \in D_X} f(i,j,k) > 0$.

Supposons $\exists x \in D_X, \exists z \in D_{\mathbf{Z}}$ t.q. $\min_{y \in D_Y} f(x,z,y) > 0$ donc $f_1 = f[X, \mathbf{Z}] \neq 0$ n'est pas une fonction nulle d'où $f(X, \mathbf{Z}, Y)$ est réductible. Il en va de même si nous supposons $\exists y \in D_Y, \exists z \in D_{\mathbf{Z}}$ t.q. $\min_{x \in D_X} f(x,z,y) > 0$. □

Définition 4.20 Si $f_1(X, \mathbf{Z})$, $f_2(\mathbf{Z}, Y)$, $\Delta(X, \mathbf{Z}, Y)$, trois fonctions de coûts (positives), est une réduction maximale de $f(X, \mathbf{Z}, Y)$ alors $\Delta(X, \mathbf{Z}, Y)$ est irréductible par rapport à X et Y.

En utilisant la propriété 4.19, nous prouvons le lemme suivant.

Lemme 4.21 *Si $f(X, \mathbf{Z}, Y)$ une fonction de coûts non nulle est* irréductible *par rapport à X et Y alors $\exists z \in D_{\mathbf{Z}}, \exists k,l \in D_Y (k \neq l), \exists u,v \in D_X (u \neq v)$ t.q. $f(u,z,k) \boxminus f(u,z,l) \not\triangleq f(v,z,k) \boxminus f(v,z,l)$.*

Preuve par l'absurde. Supposons $\forall z \in D_{\mathbf{Z}}, \forall k,l \in D_Y (k \neq l), \forall u,v \in D_X (u \neq v)$ tel que $f(u,z,k) \boxminus f(u,z,l) \triangleq f(v,z,k) \boxminus f(v,z,l)$. Nous rappelons que les fonctions de coûts ont leur coût dans $E^+ = \mathbb{N} \cup \{\top\}$ et non dans E. Nous avons :
Premier cas : $\exists u,z,k,l$ tels que $f(u,z,k) \boxminus f(u,z,l) \not\triangleq 0$
$\boxed{f(u,z,k) \boxminus f(u,z,l) > 0 \text{ ou } f(u,z,k) \boxminus f(u,z,l) = \top}$
Donc $0 \leqslant f(u,z,l) < f(u,z,k)$.
De plus $\forall v \in D_X$, $f(u,z,k) \boxminus f(u,z,l) \triangleq f(v,z,k) \boxminus f(v,z,l)$. Alors $\forall v \in D_X$, $f(v,z,k) \boxminus f(v,z,l) > 0$, ou $f(v,z,k) \boxminus f(v,z,l) = \top$, ou $f(v,z,k) \boxminus f(v,z,l) = \Omega$. Par conséquent $0 \leqslant f(v,z,l) < f(v,z,k)$ ou $f(v,z,k) = f(v,z,l) = \top$. Donc $\min_{v \in D_X} f(v,z,k) > 0$.

$\boxed{f(u,z,k) \boxminus f(u,z,l) < 0 \text{ ou } f(u,z,k) \boxminus f(u,z,l) = -\top}$
Donc $0 \leqslant f(u,z,k) < f(u,z,l)$.
De plus $\forall v \in D_X$, $f(u,z,k) \boxminus f(u,z,l) \triangleq f(v,z,k) \boxminus f(v,z,l)$.
Ainsi $\forall v \in D_X$, $f(v,z,k) \boxminus f(v,z,l) < 0$, ou $f(v,z,k) \boxminus f(v,z,l) = -\top$, ou $f(v,z,k) \boxminus f(v,z,l) = \Omega$.
Par conséquent $0 \leqslant f(v,z,k) < f(v,z,l)$ ou $f(v,z,k) = f(v,z,l) = \top$.
Donc $\min_{v \in D_X} f(v,z,l) > 0$.
Second cas : $\forall u,z,k,l$ $f(u,z,k) \boxminus f(u,z,l) \triangleq 0$
$\boxed{f(u,z,k) \boxminus f(u,z,l) = \Omega}$
Donc $f(u,z,k) = f(u,z,l) = \top$.
De plus $\forall p \in D_Y$, $f(u,z,p) \boxminus f(u,z,l) = 0$ (impossible car $f(u,z,l) = \top$) $f(u,z,p) \boxminus f(u,z,l) = f(u,z,p) \boxminus \top = \Omega$.
Ainsi $f(u,z,p) = \top$. Donc $f(u,z,l) = f(u,z,k) = f(u,z,p) = \top$, et $\min_{p \in D_Y} f(u,z,p) = \top > 0$.

$\boxed{f(u,z,k) = f(u,z,l) > 0}$
Donc $f(u,z,k) \boxminus f(u,z,l) = 0$ et par hypothèse, $\forall p \in D_Y$, $f(u,z,p) \boxminus f(u,z,l) = 0$.
Ainsi $f(u,z,k) \boxminus f(u,z,l) = f(u,z,p) \boxminus f(u,z,l)$ et $0 < f(u,z,l) = f(u,z,k) = f(u,z,p)$.
Donc $\min\limits_{p \in D_Y} f(u,z,p) > 0$.

$\boxed{f(u,z,k) = f(u,z,l) = 0}$
Par hypothèse, $\forall p \in D_Y$, $f(u,z,p) \boxminus f(u,z,l) \stackrel{\circ}{=} 0$. Alors $f(u,z,p) \boxminus 0 \stackrel{\circ}{=} 0$. Ainsi $f(u,z,p) \boxminus f(u,z,l) \neq \Omega$ et donc $f(u,z,k) \boxminus f(u,z,l) = f(u,z,p) \boxminus f(u,z,l)$.
Finalement $\forall p \in D_Y, f(u,z,p) = 0$.

Dans tous ces cas, nous pouvons conclure en utilisant la propriété 4.19 que $f(X,\mathbf{Z},Y)$ est réductible ou égale à la fonction nulle (si de manière récursive on est toujours dans le dernier sous-cas). □

Propriété 4.22 *Soit $f(X,\mathbf{Z},Y)$ maximalement réduite par $f_1(X,\mathbf{Z})$, $f_2(\mathbf{Z},Y)$ et $\Delta(X,\mathbf{Z},Y)$. Si $\Delta(u,\mathbf{z},k) \boxminus \Delta(u,\mathbf{z},l) \not\stackrel{\circ}{=} \Delta(v,\mathbf{z},k) \boxminus \Delta(v,\mathbf{z},l)$ alors*

1. *$\Delta(u,\mathbf{z},k) \boxminus \Delta(u,\mathbf{z},l) \neq \Omega$ et $\Delta(v,\mathbf{z},k) \boxminus \Delta(v,\mathbf{z},l) \neq \Omega$*
2. *$f(u,\mathbf{z},k) \boxminus f(u,\mathbf{z},l) \neq \Omega$ et $f(v,\mathbf{z},k) \boxminus f(v,\mathbf{z},l) \neq \Omega$*

Justifications.
- Si $\Delta(u,\mathbf{z},k) \boxminus \Delta(u,\mathbf{z},l) = \Omega$ alors $\Delta(u,\mathbf{z},k) \boxminus \Delta(u,\mathbf{z},l) \stackrel{\circ}{=} \Omega$ c'est en contradiction avec $\Delta(u,\mathbf{z},k) \boxminus \Delta(u,\mathbf{z},l) \not\stackrel{\circ}{=} \Delta(v,\mathbf{z},k) \boxminus \Delta(v,\mathbf{z},l)$
- Si $f(u,\mathbf{z},k) = f(u,\mathbf{z},l) = \Omega$ alors $f(u,\mathbf{z},k) \boxminus f(u,\mathbf{z},l) = \top$ nous avons alors $f(u,\mathbf{z},k) = f_1(u,\mathbf{z}) \oplus f_2(\mathbf{z},k) \oplus \Delta(u,\mathbf{z},k) = \top$ $\Delta(u,\mathbf{z},k) = f(u,\mathbf{z},k) \ominus (f_1(u,\mathbf{z}) \oplus f_2(\mathbf{z},k))$ $\Delta(u,\mathbf{z},k) = \top \ominus (f_1(u,\mathbf{z}) \oplus f_2(\mathbf{z},k)) = \top$ avec le même raisonnement nous avons $\Delta(u,\mathbf{z},l) = \top$
  de même $f(v,\mathbf{z},k) = f(v,\mathbf{z},l) = \Omega$ implique $\Delta(v,\mathbf{z},k) = \Delta(v,\mathbf{z},l) = \top$
  d'où $\Delta(u,\mathbf{z},k) \boxminus \Delta(u,\mathbf{z},l) = \Omega = \Delta(v,\mathbf{z},k) \boxminus \Delta(v,\mathbf{z},l)$ qui est en contradiction avec notre hypothèse $\Delta(u,\mathbf{z},k) \boxminus \Delta(u,\mathbf{z},l) \not\stackrel{\circ}{=} \Delta(v,\mathbf{z},k) \boxminus \Delta(v,\mathbf{z},l)$.

□

A partir de ces propriétés nous pouvons énoncer et démontrer le théorème de décomposabilité pour les fonctions de coûts quelconques suivant.

Théorème 4.23 ***Décomposabilité d'une fonction de coûts***
Une fonction de coûts $f(X,\mathbf{Z},Y)$ est décomposable par paire par rapport à X,Y si et seulement si $\forall z \in D_{\mathbf{Z}}, \forall k,l \in D_Y (k < l)$ [7], $\forall u,v \in D_{\mathbf{X}}$:

$$f(u,z,k) \boxminus f(u,z,l) \stackrel{\circ}{=} f(v,z,k) \boxminus f(v,z,l)$$

7. Chaque élément de E admet un oppposé donc $-(f(u,z,k) \boxminus f(u,z,l)) = f(u,z,l) \boxminus f(u,z,k)$ d'où la condition de tester que dans un sens les égalités en prenant un ordre arbitraire sur les valeurs

Démonstration.

$\boxed{\Longrightarrow}$ Supposons $f(X,\mathbf{Z},Y) = f_1(X,\mathbf{Z}) + f_2(\mathbf{Z},Y)$ et $f_1(X,\mathbf{Z})$, $f_2(\mathbf{Z},Y)$ fonctions positives ($0 \leqslant f_1(X,\mathbf{Z}) \leqslant f(X,\mathbf{Z},Y)$). Alors les systèmes suivants $\mathcal{S}_\mathbf{z}$ d'équations linéaires (utilisant $\boxplus$ et $\hat{=}$ operateurs) ont une solution.

$$(\mathcal{S}_\mathbf{z}) \begin{cases} f_1(1,\mathbf{z}) \boxplus f_2(\mathbf{z},1) \hat{=} f(1,\mathbf{z},1) \\ f_1(x,\mathbf{z}) \boxplus f_2(\mathbf{z},y) \hat{=} f(x,\mathbf{z},y) \quad \forall x,y \\ f_1(d_X,\mathbf{z}) \boxplus f_2(\mathbf{z},d_Y) \hat{=} f(d_X,\mathbf{z},d_Y) \end{cases}$$

$\forall \mathbf{z} \in D_\mathbf{Z}$, $\forall k,l \in D_Y$, $\forall u \in D_X$, de $\mathcal{S}_\mathbf{z}$, nous déduisons

$$\begin{aligned} f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) &\hat{=} (f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}k)) \boxminus (f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}l)) \\ &\hat{=} f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}k) \boxplus (-f_1(u\mathbf{z})) \boxplus (-f_2(\mathbf{z}l)) \\ &\hat{=} f_1(u\mathbf{z}) \boxplus (-f_1(u\mathbf{z})) \boxplus f_2(\mathbf{z}k) \boxplus (-f_2(\mathbf{z}l)) \\ &\hat{=} (f_1(u\mathbf{z}) \boxminus f_1(u\mathbf{z})) \boxplus (f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l)) \end{aligned}$$

1^{er} **cas :** $f_1(u\mathbf{z}) \in \mathbb{N}$

$$\begin{aligned} f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) &\hat{=} (f_1(u\mathbf{z}) \boxminus f_1(u\mathbf{z})) \boxplus (f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l)) \\ &\hat{=} 0 \boxplus (f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l)) \\ &\hat{=} f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l) \end{aligned}$$

2^{nd} **cas :** $f_1(u\mathbf{z}) = \top$

$$\begin{aligned} f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) &\hat{=} (f_1(u\mathbf{z}) \boxminus f_1(u\mathbf{z})) \boxplus (f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l)) \\ &\hat{=} \Omega \boxplus (f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l)) \\ &\hat{=} \Omega \end{aligned}$$

Nous pouvons conclure que $\forall \mathbf{z} \in D_\mathbf{Z}, \forall k,l \in D_Y$ $(k \neq l)$ $\forall u,v \in D_X f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) \hat{=} f(v\mathbf{z}k) \boxminus f(v\mathbf{z}l)$

$\boxed{\Longleftarrow}$ Supposons $\forall \mathbf{z} \in D_\mathbf{Z}, \forall k,l \in D_Y$ $(k < l), \forall u,v \in D_\mathbf{X} : f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) \hat{=} f(v\mathbf{z}k) \boxminus f(v\mathbf{z}l)$.
De plus, nous avons $f(X,\mathbf{Z},Y) = f_1(X,\mathbf{Z}) + f_2(\mathbf{Z},Y) + \Delta(X,\mathbf{Z},Y)$ avec $f_1(X,\mathbf{Z})$, $f_2(\mathbf{Z},Y)$ et $\Delta(X,\mathbf{Z},Y)$ des fonctions positives qui définissent les systèmes linéaires suivants :

$$(\mathcal{S}'_\mathbf{z}) \begin{cases} f_1(1\mathbf{z}) \boxplus f_2(\mathbf{z}1) \boxplus \Delta(1\mathbf{z}1) \hat{=} f(1\mathbf{z}1) \\ f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}k) \boxplus \Delta(u\mathbf{z}k) \hat{=} f(u\mathbf{z}k) \quad \forall u,k \\ f_1(d_X\mathbf{z}) \boxplus f_2(\mathbf{z}d_Y) \boxplus \Delta(d_X\mathbf{z}d_Y) \hat{=} f(d_X\mathbf{z}d_Y) \end{cases}$$

$$\begin{aligned} f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) &\hat{=} (f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}k) \boxplus \Delta(u\mathbf{z}k)) \boxminus (f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}l) \boxplus \Delta(u\mathbf{z}l)) \\ &\hat{=} f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}k) \boxplus \Delta(u\mathbf{z}k) \boxplus (-f_1(u\mathbf{z})) \boxplus (-f_2(\mathbf{z}l)) \boxplus (-\Delta(u\mathbf{z}1)) \\ &\hat{=} f_1(u\mathbf{z}) \boxplus (-f_1(u\mathbf{z})) \boxplus f_2(\mathbf{z}k) \boxplus (-f_2(\mathbf{z}l)) \boxplus \Delta(u\mathbf{z}k) \boxplus (-\Delta(u\mathbf{z}l)) \\ &\hat{=} (f_1(u\mathbf{z}) \boxminus f_1(u\mathbf{z})) \boxplus (f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l)) \boxplus (\Delta(u\mathbf{z}k) \boxminus \Delta(u\mathbf{z}l)) \end{aligned}$$

Supposons que $f(X,\mathbf{Z},Y)$ ne se décompose pas en $\{f_1(X,\mathbf{Z}), f_2(\mathbf{Z},Y)\}$, avec la propriété 4.20 nous

pouvons trouver $\Delta(X, \mathbf{Z}, Y)$ tel que $\Delta(X, \mathbf{Z}, Y)$ est une fonction de coût non nulle irréductible. Avec le lemme 4.21 nous avons $\exists\, \mathbf{z} \in D_{\mathbf{Z}}$, $k, l \in D_Y, k < l$ et $u, v \in D_X, u < v$ t.q. $\Delta(u\mathbf{z}k) \boxminus \Delta(u\mathbf{z}l) \not\doteq \Delta(v\mathbf{z}k) \boxminus \Delta(v\mathbf{z}l)$.

Nous en déduisons que $\Delta(u\mathbf{z}k) \boxminus \Delta(u\mathbf{z}l) \neq \Omega$ et $\Delta(v\mathbf{z}k) \boxminus \Delta(v\mathbf{z}l) \neq \Omega$ avec la définiton de $\stackrel{\circ}{=}$.
Par la propriété 4.22 nous avons également $f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) \neq \Omega \neq f(v\mathbf{z}k) \boxminus f(v\mathbf{z}l)$ Alors $f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) \stackrel{\circ}{=} f(v\mathbf{z}k) \boxminus f(v\mathbf{z}l) \stackrel{\circ}{=} \varphi$ t.q. $\varphi \in \mathbb{Z} \cup \{\top, -\top\}$.

La propriété 4.22 nous permet également de déduire [8] $f_1(u\mathbf{z}) \boxminus f_1(u\mathbf{z}) = 0 = f_1(v\mathbf{z}) \boxminus f_1(v\mathbf{z})$ et $f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l) \neq \Omega$.

En utilisant les propriétés précédentes :

$$\begin{array}{rcl}
\Delta(u\mathbf{z}k) \boxminus \Delta(u\mathbf{z}l) & \neq & \Delta(v\mathbf{z}k) \boxminus \Delta(v\mathbf{z}l) \\
f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l) \boxplus \Delta(u\mathbf{z}k) \boxminus \Delta(u\mathbf{z}l) & \neq & f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l) \boxplus \Delta(v\mathbf{z}k) \boxminus \Delta(v\mathbf{z}l) \\
f_1(u\mathbf{z}) \boxminus f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l) \boxplus \Delta(u\mathbf{z}k) \boxminus \Delta(u\mathbf{z}l) & \neq & f_1(v\mathbf{z}) \boxminus f_1(v\mathbf{z}) \boxplus f_2(\mathbf{z}k) \boxminus f_2(\mathbf{z}l) \boxplus \Delta(v\mathbf{z}k) \boxminus \Delta(v\mathbf{z}l) \\
f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}k) \boxplus \Delta(u\mathbf{z}k) \boxminus (f_1(u\mathbf{z}) \boxplus f_2(\mathbf{z}l) \boxplus \Delta(u\mathbf{z}l)) & \neq & f_1(v\mathbf{z}) \boxplus f_2(\mathbf{z}k) \boxplus \Delta(v\mathbf{z}k) \boxminus (f_1(v\mathbf{z}) \boxplus f_2(\mathbf{z}l) \boxplus \Delta(v\mathbf{z}l)) \\
f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) & \neq & f(v\mathbf{z}k) \boxminus f(v\mathbf{z}l)
\end{array}$$

C'est en contradiction avec la première hypothèse : $\forall\, k, l \in D_Y$, $\forall\, \mathbf{z} \in D_{\mathbf{Z}}$

$$f(u\mathbf{z}k) \boxminus f(u\mathbf{z}l) \stackrel{\circ}{=} f(v\mathbf{z}k) \boxminus f(v\mathbf{z}l)$$

□

Remarque 4.24 *Nous pouvons réduire le nombre d'égalités à vérifier dans la pratique, puisque* $f(u, z, k) \boxminus f(u, z, l) = \Omega \stackrel{\circ}{=} a$, $\forall a \in E$, *il suffit de trouver une valeur* $u \in D_{\mathbf{X}}$ *telle que* $f(u, z, k) \boxminus f(u, z, l) \neq \Omega$ *et de tester* $\forall v \in D_{\mathbf{X}}$ $(v \neq u)$ *telle que* $f(v, z, k) \boxminus f(v, z, l) \neq \Omega$
$f(u, z, k) \boxminus f(u, z, l) = f(v, z, k) \boxminus f(v, z, l)$
Puisque l'égalité est transitive le théorème 4.23 est bien vérifié.

Exemple 4.25 *Si nous reprenons la fonction* f *de l'exemple 4.10 page 63, elle est bien décomposable par rapport à* A, C *car :*

A	B	C	f
0	0	0	$\top$
0	0	1	$\top$
0	1	0	4
0	1	1	1
1	0	0	3
1	0	1	2
1	1	0	7
1	1	1	4

$$\begin{array}{lcl|lcl|lcl}
f(0,0,0) \boxminus f(0,0,1) & \stackrel{\circ}{=} & f(1,0,0) \boxminus f(1,0,1) & \top \boxminus \top & \stackrel{\circ}{=} & 3 \boxminus 2 & \Omega & \stackrel{\circ}{=} & 1 \quad \checkmark \\
f(0,1,0) \boxminus f(0,1,1) & \stackrel{\circ}{=} & f(1,1,0) \boxminus f(1,1,1) & 4 \boxminus 1 & \stackrel{\circ}{=} & 7 \boxminus 4 & 3 & \stackrel{\circ}{=} & 3 \quad \checkmark
\end{array}$$

8. $f(u, \mathbf{z}, k) \boxminus f(u, \mathbf{z}, l) \neq \Omega$ implique que $f(u, \mathbf{z}, k) \neq \top$ ou $f(u, \mathbf{z}, l) \neq \top$ d'où $f_1(u, \mathbf{z}) < \top$ et $f_2(\mathbf{z}, k) \neq \top$ ou $f_2(\mathbf{z}, l) \neq \top$.

Le théorème de décomposabilité correspond à un test d'existence des deux fonctions $f_1(X, \mathbf{Z})$ et $f_2(\mathbf{Z}, Y)$, le théorème suivant montre comment les construire en utilisant les projections et les soustractions classiques des fonctions de coûts.

4.2.3 Construction de la décomposition par paire et propriétés

4.2.3.1 Construction des deux nouvelles fonctions

Théorème 4.26 *Soit $f(X, \mathbf{Z}, Y)$ décomposable par paire par rapport à X, Y. Alors $f_1(X, \mathbf{Z}) = f[X, \mathbf{Z}]$ et $f_2(\mathbf{Z}, Y) = (f \ominus f_1)[\mathbf{Z}, Y]$ est une décomposition valide de f, i.e. $f = f_1 \oplus f_2$.*

Ce théorème se démontre à l'aide du lemme suivant.

Lemme 4.27 *Soit une fonction de coûts $f(X, \mathbf{Z}, Y)$ décomposable par paire par rapport à X, Y. Si $\forall x \in D_X,\ f_1(x, \mathbf{z}) = \min_{y \in D_Y} f(x, \mathbf{z}, y)$ alors $\forall \mathbf{z} \in D_{\mathbf{Z}}\ \exists k \in D_Y$ tel que $\forall x \in D_X,\ f_1(x, \mathbf{z}) = f(x, \mathbf{z}, k)$*
.

Démonstration. Soit $z \in D_{\mathbf{Z}}$.

Premier cas : $\boxed{\exists u \in D_X,\ \exists y,\ f(u, \mathbf{z}, y) \neq \top}$

Soit $D_Y = \{k_1, k_2, \ldots, k_{d_Y}\}$ tel que $\forall i < j,\ f(u, \mathbf{z}, k_i) \leqslant f(u, \mathbf{z}, k_j)$. Nous en déduisons $\forall i \in [1, d_Y],\ f(u, \mathbf{z}, k_1) \leqslant f(u, \mathbf{z}, k_i)$ et $f_1(u, \mathbf{z}) = \min_{y \in D_Y} f(u, \mathbf{z}, y) = f(u, \mathbf{z}, k_1)$.
En utilisant le théorème 4.23 page 69, nous trouvons $\forall k_p \in D_K\ \forall x \in D_X,\ 0 \leqslant f(u, \mathbf{z}, k_p) \boxminus f(u, \mathbf{z}, k_1) \triangleq f(x, \mathbf{z}, k_p) \boxminus f(x, \mathbf{z}, k_1)$. Donc $f(x, \mathbf{z}, k_1) \leqslant f(x, \mathbf{z}, k_p)$ ou $f(x, \mathbf{z}, k_1) = f(x, \mathbf{z}, k_p) = \top$, ainsi $\forall x \in D_X, f_1(x, \mathbf{z}) = \min_{y \in D_Y} f(x, \mathbf{z}, y) = f(x, \mathbf{z}, k_1)$.

Second cas : $\boxed{\forall x \in D_x,\ \forall y,\ f(x, z, y) = \top}$

$\forall x \in D_X,\ f_1(x, z) = \min_{y \in D_Y} f(x, z, y) = \top$, donc soit $k \in D_y$, $\forall x \in D_X,\ f_1(x, z) = f(x, z, k) = \top$

□

Démonstration du théorème 4.26. Nous avons $f_1(x, z) = \min_{y \in D_Y} f(x, z, y)$ et $f_2(z, y) = \min_{x \in D_X} [f(x, z, y) \ominus f_1(x, z)]$ donc $f_2(z, y) \leqslant f(x, z, y) \ominus f_1(x, z)$.

Si $f_2(z, y) = f(x, z, y) \ominus f_1(x, z)$ alors $f(x, z, y) = f_1(x, z) \oplus f_2(z, y)$.

Si $f_2(z, y) < f(x, z, y) \ominus f_1(x, z)$,

$\boxed{f(x, z, y) = f_1(x, z) = \top}$
Nous avons $f_1(x, z) \oplus \min_{x \in D_X} [f(x, z, y) \ominus f_1(x, z)] = \top = f(x, z, y)$
donc $f(x, z, y) = f_1(x, z) \oplus f_2(z, y)$.

$\boxed{f(x, z, y) \neq \top \textbf{ ou } f_1(x, z) \neq \top}$

Nous avons $\min_{u \in D_X} [f(u,z,y) \ominus f_1(u,z)] < f(x,z,y) \ominus f_1(x,z)$ donc $f(u,z,y) \ominus f_1(u,z) < f(x,z,y) \ominus f_1(x,z)$ avec $u = \text{argmin}_{u \in D_X}[f(u,z,y) \ominus f_1(u,z)]$.
En utilisant le lemme 4.27, nous avons $l = \underset{k \in D_Y}{\text{argmin}}\, f(x,z,k) = \underset{k \in D_Y}{\text{argmin}}\, f(u,z,k)$, ainsi $f(u,z,y) \ominus f(u,z,l) < f(x,z,y) \ominus f(x,z,l)$, mais également $f(u,z,y) \boxminus f(u,z,l) \stackrel{\circ}{=} f(x,z,y) \boxminus f(x,z,l)$ car f est décomposable par paire par rapport à X, Y.
De plus, $f(x,z,y) \neq \top$ ou $f(x,z,l) \neq \top$, ainsi $f(u,z,y) \ominus f(u,z,l) = f(x,z,y) \ominus f(x,z,l)$.
Finalement $f_2(z,y) = f(x,z,y) \ominus f_1(x,z)$, ainsi $f(x,z,y) = f_1(x,z) \oplus f_2(z,y)$. □

Exemple 4.28 *Les fonctions proposées dans les exemples 4.9 et 4.10 page 63 ont été construites à l'aide de ce théorème.*

4.2.3.2 Complexités

Propriété 4.29 ***Complexités***
La complexité dans le pire des cas de l'application de la décomposition par paire pour e fonctions de coûts originales avec une arité r maximale est $O(er^3d^{r+1})$ en temps et $O(ed^r)$ en espace, soit linéaire par rapport à la taille du probléme.

Justifications. Appliquer la décomposition par paire sur une fonction de coûts $f(\mathbf{S})$ avec $r = |\mathbf{S}|$ peut nécessiter $\frac{r(r-1)}{2}$ tests, *i.e.* vérifier les égalités pour toutes les paires de variables de $\mathbf{S}$. Et il peut produire des fonctions (non vides) d'arité (r-1). La répétition de la décomposition par paire pour les fonctions de coûts résultantes produira au plus $\min(\binom{r-p}{r}, 2^p)$ fonctions de coûts d'arité $(r-p)$ avec des portées différentes. Chaque test est linéaire par rapport à la taille des fonctions de coûts $(r-p)$-aire en $O(d^{(r-p)+1})$. Donc la complexité globale dans le pire des cas de l'application de la décomposition de paire pour e fonctions de coûts originales avec une arité r maximale est $O(e\sum_{p=0}^{r-1} 2^p(r-p)^2 d^{(r-p)+1}) = O(e\sum_{p=0}^{r-1}(r-p)^2 d^{r+1}) = O(e\frac{r(r+1)(2r+1)}{6}d^{r+1}) = O(er^3d^{r+1})$ en temps et $O(e\max_{p=0}^{r-1} 2^p d^{(r-p)}) = O(ed^r)$ en espace. □

4.2.3.3 Propriétés

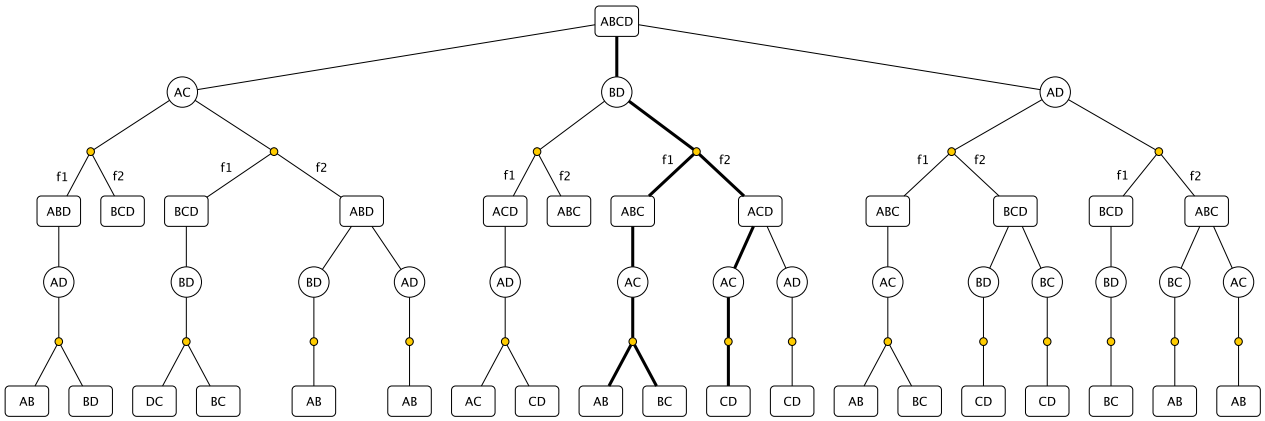
Les fonctions peuvent être décomposables par rapport à différentes paires de variables, par exemple la fonction $f(A,B,C,D)$ de la figure 4.3a est décomposable par rapport à $\{A,C\}$ mais également par rapport à $\{A,D\}$ et $\{B,D\}$. Dans le cas de la décomposition par rapport à A et C, nous obtiendrons deux fonctions $g(A,B,D)$ et $h(B,C,D)$ (*i.e.* $f = g \oplus h$). Pour construire ces deux fonctions en utilisant le théorème 4.26 nous pouvons définir (1) $f_1 = g$ et $f_2 = h$ ou (2) $f_1 = h$ et $f_2 = g$, les tables 4.1 et 4.2 page 76 illustrent l'application du théorème dans ces deux cas. Le choix de la fonction f_1 influe sur le contenu des tables mais également sur leur décomposabilité. En effet, la fonction $h(B,C,D)$

est décomposable dans le cas (1) (*cf.* table 4.1b page 76) mais ne l'est dans le cas (2) (*cf.* table 4.2b page 76).

A chaque étape, le choix des variables utilisées pour la décomposition et la manière de répartir les coûts dans f_1 et f_2 peuvent influencer la décomposabilité de f_1 et f_2 à l'itération suivante. Par exemple, la figure 4.3b représente toutes les possibilités de décomposition de la fonction de coûts donnée par la figure 4.3a.

A	B	C	D	$f(A,B,C,D)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	5
0	0	1	1	4
0	1	0	0	6
0	1	0	1	6
0	1	1	0	4
0	1	1	1	3
1	0	0	0	0
1	0	0	1	0
1	0	1	0	5
1	0	1	1	4
1	1	0	0	9
1	1	0	1	9
1	1	1	0	7
1	1	1	1	6

(a) Une fonction de coûts



(b) Représentation de toutes les décompositions possibles de la fonction

Figure 4.3 – Trois types de nœuds sont à identifier dans la figure (b) : les nœuds rectangulaires pour identifier les fonctions, les nœuds ronds avec label pour identifier par rapport à quelles variables la décomposition par paire est réalisée et enfin les nœuds ronds sans label pour symboliser la construction des deux fonctions résultantes de la décomposition. Le chemin en gras représente les décompositions par paire réalisées avec l'ordre DAC : $A < B < C < D$.

$f_1 = g(A,B,D)$		D	
		0	1
A	B		
0	0	0	0
	1	4	3
1	0	0	0
	1	7	6

(a)

$f_2 = h(B,C,D)$		D	
		0	1
B	C		
0	0	0	0
	1	5	4
1	0	2	3
	1	0	0

(b)

Table 4.1 – Application du théorème 4.26 sur la fonction $f(A,B,C,D)$ de la figure 4.3a avec $f_1 = g(A,B,D)$ et $f_2 = h(B,C,D)$. (La fonction $h(B,C,D)$ n'est pas décomposable.)

$f_2 = g(A,B,D)$		D	
		0	1
A	B		
0	0	0	0
	1	0	0
1	0	0	0
	1	3	3

(a)

$f_1 = h(B,C,D)$		D	
		0	1
B	C		
0	0	0	0
	1	5	4
1	0	6	6
	1	4	3

(b)

Table 4.2 – Application du théorème 4.26 sur la fonction $f(A,B,C,D)$ de la figure 4.3a avec $f_1 = g(A,B,D)$ et $f_2 = h(B,C,D)$. (Les deux fonctions ternaires sont encore décomposables.)

Sous certaines conditions, il est possible de trouver une séquence de décomposition de paire c'est-à-dire la paire de variables à sélectionner dans f et comment distribuer f dans f_1 et f_2 qui mène à une décomposition optimale (avec des arités minimales).

Propriété 4.30 ***Décomposition minimale unique pour des fonctions de coûts finis [Hammersley and Clifford, 1971]***
Une fonction de coûts avec seulement des coûts finis admet une décomposition unique et minimale.

Jusfications. Si une fonction de coûts $f(\mathbf{S})$ a uniquement des coûts finis, alors sa distribution de probabilité associée $\mathbb{P}_f$ (voir Théorème 4.7 page 60) est strictement positive. Le théorème d'*Hammersley & Clifford* [Hammersley and Clifford, 1971] [9] peut s'appliquer. Cette factorisation, réalisée sur $\mathbb{P}_f$, est équivalente à une somme de fonctions de coûts en prenant $-\log(\mathbb{P}_f)$ (voir la preuve du Théorème 4.7). De cette décomposition résultante, il est facile de construire une séquence inverse de fonctions de coûts (positives) valides jusqu'à atteindre la fonction de coûts f originale.

□

9. Voir aussi le Théorème 4.5 page 121 dans [Koller and Friedman, 2009].

Cependant, dès qu'il y a des coûts infinis, le théorème précédent ne peut pas s'appliquer.

Dans ce cas, nous proposons une approche heuristique dont le but est d'accumuler les coûts sur les premières variables dans l'ordre des variables de DAC. Cela devrait améliorer les bornes inférieures basées sur DAC. Nous testons donc des paires de variables dans $f(\mathbf{S})$ dans l'ordre DAC inverse [10] et projetons les fonctions de coûts (Théorème 4.26) sur la portée contenant les premières variables dans l'ordre DAC en premier. Dans l'exemple 4.32, supposons l'ordre (A,B,C,D), cela correspond à projeter en premier sur $\{A, B, C\}$. Nous montrons que notre approche heuristique est capable de trouver une décomposition optimale pour quelques fonctions particulières.

Dans la section 4.1 nous avons vu que certaines contraintes dures pouvaient se décomposer en un arbre de contraintes binaires, notre décomposition par paire est capable de les identifier.

Propriété 4.31 ***Identification de structure d'arbre pour les contraintes dures [Meiri et al., 1990]***
Une fonction de coûts avec seulement des coûts infinis qui admet une décomposition en arbre de contraintes binaires est identifiable par notre stratégie de décomposition.

Ce résultat vient de [Meiri et al., 1990] et le fait qu'une décomposition par paire par rapport à X, Y supprimera une arête redondante $\{X, Y\}$ dans le réseau minimal [11] sans affecter l'ensemble des solutions. Par exemple, une contrainte globale d'égalité sur $\mathbf{S}$ sera décomposée en un arbre de contraintes d'égalité binaires : $f(\mathbf{S}) = \sum_{Y \in \mathbf{S} \setminus \{X\}} f_Y(X, Y)$ avec $f_Y(X, Y) \equiv (X = Y)$.

De la même façon, une fonction de coûts linéaire $f(X_1, \ldots, X_r) = \sum_{i=1}^{r} a_i X_i$ peut être décomposée en une somme de r fonctions de coûts unaires [12].

4.2.4 Projections et Soustractions sur les fonctions de coûts binaires

Quand un WCSP est décomposé par paire, il est tout de même possible d'inférer une information à partir des fonctions de coûts non binaires par projection et soustraction sur des fonctions de coûts de plus petites arités. Nous choisissons de projeter chaque fonction de coûts non binaire sur toutes les fonctions de coûts binaires possibles étant donnée la portée de la fonction : le *binProject*. Il est fait en suivant un ordre des projections et des soustractions compatible avec l'ordre des variables DAC (*i.e.* on projette d'abord sur les paires de variables avec les plus petites positions dans l'ordre DAC). Chaque projection $f_i(X, Y) = f[X, Y]$ est suivie par une soustraction $f = f \ominus f_i$ afin de préserver l'équivalence du problème. Notons que f peut être vide après ces soustractions et est alors supprimée du problème. Notons aussi que la même fonction binaire peut recevoir des projections de coûts de plusieurs fonctions non binaires ayant des portées se chevauchant, résultant d'une plus forte inférence.

10. Nous testons C, D avant A, B si $\max(C, D) > \max(A, B) \vee (\max(C, D) = \max(A, B) \wedge \min(C, D) > \min(A, B))$.
11. Dans le cadre des CSP, le CSP binaire, appelé *réseau minimal*, qui est la meilleure approximation d'une relation non binaire, est défini par les projections de la relation sur toutes les paires de variables [Meiri et al., 1990].
12. Notons que l'arc cohérence souple ferait de même si la fonction de coûts vide résultante est supprimée après les transformations de projection.

La complexité dans le pire des cas de *binProject* appliquée sur m fonctions de coûts avec r comme arité maximale est $O(mr^2d^r)$ en temps et $O(mr^2d^2)$ en espace.

Exemple 4.32 *Cet exemple déroule l'ensemble des propriétés vues dans ce chapitre : le test de décomposabilité, la construction des fonctions et l'application du* binProject.

Soit une fonction de coûts quaternaire $f(A, B, C, D)$, l'ordre DAC $A < B < C < D$.

A	B	C	D	f
0	0	0	0	5
0	0	0	1	$\top$
0	0	1	0	$\top$
0	0	1	1	3
0	1	0	0	6
0	1	0	1	5
0	1	1	0	$\top$
0	1	1	1	$\top$
1	0	0	0	4
1	0	0	1	$\top$
1	0	1	0	$\top$
1	0	1	1	2
1	1	0	0	2
1	1	0	1	1
1	1	1	0	3
1	1	1	1	1

Testons la décomposabilité de f par rapport à A et D :

$$f(0,0,0,0) \boxminus f(0,0,0,1) \;\hat{=}\; f(1,0,0,0) \boxminus f(1,0,0,1) \quad | \quad 5 \boxminus \top \;\hat{=}\; 4 \boxminus \top \quad \checkmark$$

$$f(0,0,1,0) \boxminus f(0,0,1,1) \;\hat{=}\; f(1,0,1,0) \boxminus f(1,0,1,1) \quad | \quad \top \boxminus 3 \;\hat{=}\; \top \boxminus 2 \quad \checkmark$$

$$f(0,1,0,0) \boxminus f(0,1,0,1) \;\hat{=}\; f(1,1,0,0) \boxminus f(1,1,0,1) \quad | \quad 6 \boxminus 5 \;\hat{=}\; 2 \boxminus 1 \quad \checkmark$$

$$f(0,1,1,0) \boxminus f(0,1,1,1) \;\hat{=}\; f(1,1,1,0) \boxminus f(1,1,1,1) \quad | \quad \top \boxminus \top \;\hat{=}\; 3 \boxminus 1 \quad \checkmark$$

Les fonctions f_1 et f_2 que nous obtenons avec le théorème de construction sont irréductibles, mais elles peuvent se décomposer en $f_1(A, B, C) = b_1(A, B)+b_2(B, C)+f_3(A, B, C)$ et $f_2(B, C, D) = b_3(B, D)+b_4(C, D)+f_4(B, C, D)$ par le binProject *en suivant l'ordre DAC.*

A	B	C	f_1
0	0	0	5
0	0	1	3
0	1	0	5
0	1	1	$\top$
1	0	0	4
1	0	1	2
1	1	0	1
1	1	1	1

$=$

A	B	b_1
0	0	3
0	1	5
1	0	2
1	1	1

$\oplus$

B	C	b_2
0	0	2
0	1	0
1	0	0
1	1	0

$\oplus$

A	B	C	f_3
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	$\top$
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

B	C	D	f_2
0	0	0	0
0	0	1	$\top$
0	1	0	$\top$
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	2
1	1	1	0

$=$

B	D	b_3
0	0	0
0	1	0
1	0	1
1	1	0

$\oplus$

C	D	b_4
0	0	0
0	1	0
1	0	1
1	1	0

$\oplus$

B	C	D	f_4
0	0	0	0
0	0	1	$\top$
0	1	0	$\top$
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

L'utilisation unique du binProject *sur la fonction f suivant l'ordre DAC précédent nous amène à projeter, dans l'ordre, sur les fonctions binaires de portée suivantes : $\{A, B\}$, $\{A, C\}$, $\{A, D\}$, $\{B, C\}$, $\{B, D\}$ et $\{C, D\}$. Les fonctions sur $\{A, C\}$, $\{B, D\}$ et $\{C, D\}$ sont des fonctions vides. Nous obtenons alors $f(A, B, C, D) = f_{AB} + f_{AC} + f_{BC} + f_{ABCD}$ avec les tables suivantes*

A	B	
0	0	3
0	1	5
1	0	2
1	1	1

A	D	
0	0	1
0	1	0
1	0	2
1	1	0

B	C	
0	0	1
0	1	0
1	0	0
1	1	0

A	B	C	D	f
0	0	0	0	0
0	0	0	1	$\top$
0	0	1	0	$\top$
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	$\top$
0	1	1	1	$\top$
1	0	0	0	0
1	0	0	1	$\top$
1	0	1	0	$\top$
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

L'application du binProject *sans l'utilisation de la décomposition nous donne trois fonctions de coûts binaires*

et une fonction quaternaire résiduelle. Alors que l'application préalable de la décomposition nous permettait d'avoir quatre fonctions de coûts binaires et deux ternaires.

L'arité maximale a été réduite après la décomposition associée au binProject *ce qui n'est pas le cas en appliquant seulement le* binProject *pour cette fonction de coût* $f(A, B, C, D)$.

4.3 Décomposition par paire approchée

Lors de la transformation des réseaux bayésiens aux Wcsp nous réalisons une approximation des valeurs réelles des tables de probabilités conditionnelles. Des tables de probabilités conditionnelles qui étaient décomposables par paire à l'origine peuvent se retrouver non décomposables après l'approximation des valeurs. De plus, les résultats obtenus pour la décomposition par paire (*cf.* section 4.4) nous encouragent à décomposer lorsque c'est possible. Bien évidemment toutes les fonctions ne sont pas décomposables, dans le cas où une approximation de l'optimum peut être intéressant, une approximation des problèmes par décompostion approchée de leurs fonctions est à explorer.

Cette section présente un travail non encore abouti, elle propose d'établir un test de décomposabilité approchée et de construire les fonctions résultantes sur le même principe que la décomposition par paire.

La définition suivante s'inspire de la décomposition approchée de l'article [Wexler and Meek, 2008]. Cette décomposition a pour but de décomposer une fonction sur le principe de la décomposition par paire dont la somme des deux fonctions résultantes admet une erreur relative par rapport à la fonction originelle paramétrée par une valeur $\varepsilon \in [0, 1]$.

Définition 4.33 **ε-décomposition par paire**
Soit $\varepsilon \in [0, 1]$, une ε-décomposition par paire d'une fonction de coût $f(S)$ par rapport à deux variables X et Y est une approximation de la fonction f définie par deux fonctions $f_1(S \setminus Y)$ et $f_2(S \setminus X)$ telle que :

$$\frac{1}{1+\varepsilon} f(\mathbf{S}) \leqslant f_1(\mathbf{S} \setminus Y) \oplus f_2(\mathbf{S} \setminus X) \leqslant (1+\varepsilon) f(\mathbf{S})$$

Remarque 4.34 *$f_1 \oplus f_2$ est un coût entier, mais $\frac{1}{1+\varepsilon} f$ et $(1+\varepsilon)f$ ne le sont pas forcément, notre inéquation peut se restreindre aux valeurs entières telles que*

$$\left\lceil \frac{1}{1+\varepsilon} f(\mathbf{S}) \right\rceil \leqslant f_1(\mathbf{S} \setminus Y) \oplus f_2(\mathbf{S} \setminus X) \leqslant \left\lfloor (1+\varepsilon) f(\mathbf{S}) \right\rfloor$$

Exemple 4.35 *Soit $\varepsilon = 0.1$, la fonction f suivante est ε-décomposable (par rapport à X et Y) et son ε-décomposition peut s'exprimer avec les fonctions $f_1(X)$ et $f_2(Y)$.*

X	Y	f
0	0	503
0	1	869
1	0	420
1	1	827

X	$f_1(X)$
0	370
1	282

Y	$f_2(Y)$
0	138
1	470

Le tableau suivant permet de vérifier que la définition est vérifée pour tous les tuples

X	Y	$\frac{1}{1+\varepsilon}f$	f_1+f_2	$(1+\varepsilon)f$
0	0	457	508	553
0	1	790	840	955
1	0	381	420	462
1	1	751	752	909

Propriété 4.36 *Un fonction f est ε-décomposable par paire par rapport à X et Y si et seulement si il existe une fonction $g(\mathbf{S})$ telle que*

1. *g soit décomposable par paire par rapport à X et Y*
2. $\frac{1}{1+\varepsilon}f(\mathbf{S}) \leqslant g(\mathbf{S}) \leqslant (1+\varepsilon)f(\mathbf{S})$

Justifications. Si f est ε-décomposable alors $g(\mathbf{S}) = f_1(\mathbf{S} \setminus Y) \oplus f_2(\mathbf{S} \setminus X)$ vérifie les deux conditions par construction.

Soit $g(\mathbf{S}) = g_1(\mathbf{S} \setminus Y) \oplus g_2(\mathbf{S} \setminus X)$ telle que $\frac{1}{1+\varepsilon}f(\mathbf{S}) \leqslant g(\mathbf{S}) \leqslant (1+\varepsilon)f(\mathbf{S})$ donc
$\frac{1}{1+\varepsilon}f(\mathbf{S}) \leqslant g_1(\mathbf{S} \setminus Y) \oplus g_2(\mathbf{S} \setminus X) \leqslant (1+\varepsilon)f(\mathbf{S})$
f est donc ε-décomposable par paire par rapport à X et Y. □

Théorème 4.37 *Si une fonction de coûts $f(X, \mathbf{Z}, Y)$ est ε-décomposable par rapport à X et Y alors $\forall \mathbf{z} \in D_{\mathbf{Z}}$ $\forall k, l \in D_Y$:*

$$\max_{u \in D_X} \left(\frac{1}{1+\varepsilon} f(u,z,k) \boxminus (1+\varepsilon) f(u,z,l) \right) \leqslant \min_{u \in D_X} \left((1+\varepsilon) f(u,z,k) \boxminus \frac{1}{1+\varepsilon} f(u,z,l) \right)$$

Démonstration. Supposons que $f(X, \mathbf{Z}, Y)$ soit ε-décomposable par rapport à X et Y. Nous avons $\forall \mathbf{z} \in D_{\mathbf{Z}}$ $\forall y \in D_Y$ et $\forall x \in D_X$:

$$\frac{1}{1+\varepsilon} f(x,\mathbf{z},y) \leqslant f_1(x,\mathbf{z}) \oplus f_2(\mathbf{z},y) \leqslant (1+\varepsilon) f(x,\mathbf{z},y) \tag{4.3}$$

Soit $u, v \in D_X$, $k, l \in D_Y$ et $\mathbf{z} \in D_{\mathbf{Z}}$

$$\begin{aligned} \frac{1}{1+\varepsilon} f(u,\mathbf{z},k) &\leqslant f_1(u,\mathbf{z}) \oplus f_2(\mathbf{z},k) \leqslant (1+\varepsilon) f(u,\mathbf{z},k) \\ \frac{1}{1+\varepsilon} f(u,\mathbf{z},l) &\leqslant f_1(u,\mathbf{z}) \oplus f_2(\mathbf{z},l) \leqslant (1+\varepsilon) f(u,\mathbf{z},l) \end{aligned} \tag{4.4}$$

$$\boxminus((1+\varepsilon) f(u,\mathbf{z},l)) \leqslant \boxminus(f_1(u,\mathbf{z}) \oplus f_2(\mathbf{z},l)) \leqslant \boxminus(\frac{1}{1+\varepsilon} f(u,\mathbf{z},l)) \tag{4.5}$$

En combinant les deux équations (4.4) et (4.5) nous obtenons l'encadrement suivant :

$$\frac{1}{1+\varepsilon}f(u,\mathbf{z},k) \boxminus (1+\varepsilon)f(u,\mathbf{z},l) \leqslant f_1(u,\mathbf{z}) \oplus f_2(\mathbf{z},k) \boxminus (f_1(u,\mathbf{z}) \oplus f_2(\mathbf{z},l))$$
$$\leqslant (1+\varepsilon)f(u,\mathbf{z},l) \boxminus (\frac{1}{1+\varepsilon}f(u,\mathbf{z},l))$$

d'où

$$\frac{1}{1+\varepsilon}f(u,\mathbf{z},k) \boxminus (1+\varepsilon)f(u,\mathbf{z},l) \leqslant f_2(\mathbf{z},k) \boxminus f_2(\mathbf{z},l) \leqslant (1+\varepsilon)f(u,\mathbf{z},l) \boxminus (\frac{1}{1+\varepsilon}f(u,\mathbf{z},l)) \quad (4.6)$$

Cet encadrement doit être vérifié pour toutes les valeurs $u \in D_X$, nous pouvons ainsi établir un encadrement de $f_2(\mathbf{z},k) \boxminus f_2(\mathbf{z},l)$ par rapport à toutes ces valeurs u :

$$\max_{u\in D_X}\left(\frac{1}{1+\varepsilon}f(u,z,k) \boxminus (1+\varepsilon)f(u,z,l)\right) \leqslant \min_{u\in D_X}\left((1+\varepsilon)f(u,z,k) \boxminus \frac{1}{1+\varepsilon}f(u,z,l)\right) \quad (4.7)$$

□

Remarque 4.38

– *Actuellement l'équivalence n'a pas été démontrée, mais une piste consisterait à construire une fonction g décomposable par paire par rapport à X et Y telle que* $\frac{1}{1+\varepsilon}f(\mathbf{S}) \leqslant g(\mathbf{S}) \leqslant (1+\varepsilon)f(\mathbf{S})$ *à partir de l'hypothèse*

$$\max_{u\in D_X}\left(\frac{1}{1+\varepsilon}f(u,z,k) \boxminus (1+\varepsilon)f(u,z,l)\right) \leqslant \min_{u\in D_X}\left((1+\varepsilon)f(u,z,k) \boxminus \frac{1}{1+\varepsilon}f(u,z,l)\right)$$

– *Lorsque* $\varepsilon = 0$*, l'équivalence est établie car*

$$\max_{u\in D_X}\left(\frac{1}{1+\varepsilon}f(u,z,k) \boxminus (1+\varepsilon)f(u,z,l)\right) \leqslant \min_{u\in D_X}\left((1+\varepsilon)f(u,z,k) \boxminus \frac{1}{1+\varepsilon}f(u,z,l)\right)$$

est en réalité

$$\max_{u\in D_X}\left(f(u,z,k) \boxminus f(u,z,l)\right) \leqslant \min_{u\in D_X}\left(f(u,z,k) \boxminus f(u,z,l)\right)$$

qui implique $\forall u,v \in D_X,\ k,l \in D_Y,$

$$f(u,z,k) \boxminus f(u,z,l) \doteq f(v,z,k) \boxminus f(v,z,l)$$

4.4 Résultats expérimentaux

DFBB-VE(i) [13] maintenant EDAC et utilisant un ordre dynamique d'affectation des variables basé sur les conflits et une pondération des fonctions de coûts [Lecoutre et al., 2009] a obtenu les meilleurs

13. TOULBAR2version 0.9.4.

résultats de l'évaluation de MPE de UAI'08 [14] (et UAI'10 [15] pour le cas à 20 minutes) excepté pour deux benchmarks difficiles : les réseaux d'analyse de liaison et les réseaux en grille.

Nous avons donc testé la décomposition par paire et le *binProject* en pré-traitement de DFBB-VE(i) sur ces deux familles de benchmarks.

Les réseaux d'analyse de liaison génétique (Linkage). Les instances ont entre 334 et 1289 variables. Le domaine maximal de taille d est entre 3 et 7 et l'arité la plus grande est 5. Cette famille de benchmarks est constituée de 22 problèmes [16]. Le problème d'analyse de liaison [Lauritzen and Sheehan, 2003] est très similaire au problème de reconstruction d'haplotypes. Ces problèmes traitent des familles d'une soixantaine d'individus avec des données sur une vingtaine de marqueurs.

Les réseaux en grille (Grids). Chaque problème est une grille $l \times l$ et chaque table de probabilités conditionnelles ternaires est générée aléatoirement et uniformément. 50, 75 ou 90% des tables sont déterministes et les variables sont Booléennes. Pour ces instances, l varie de 12 à 50 (*i.e.* $n = 2500$ variables). Cette famille de benchmarks a 32 problèmes[16].

Les expérimentations ont été réalisées sur un ordinateur 2.6 GHz Intel Xeon avec 4GB sous Linux 2.6. Les temps CPU totaux sont en secondes et limités à une heure pour chaque instance ("-" représente un dépassement de temps). Aucune borne supérieure n'est donnée au départ. Nous testons l'application de la décomposition par paire (DFBB-VE(i)+dec), de binProject sur les fonctions de coûts n-aires (DFBB-VE(i)+bP), et la combinaison des deux techniques (DFBB-VE(i)+dec+bP). Toutes les méthodes utilisent l'heuristique *MinFill* (*cf.* . page 14) pour l'ordre d'élimination de variables et l'ordre inverse est utilisé pour DAC.

La Table 4.3 donne le nombre de problèmes résolus par les différentes versions de DFBB-VE(i).

La décomposition par paire comme le binProject permet de résoudre plus de problèmes que DFBB-VE(i) seul. La combinaison de la décomposition et du binProject permet d'obtenir les meilleurs résultats (sauf pour les Linkage avec $i = 2, 3$), elle permet également d'utiliser un degré d'élimination de variables plus important que pour les autres méthodes avant de voir le nombre d'instances résolues se détériorer. Remarquablement, toutes les instances de grilles sont résolues avec de l'élimination de variables bornée pour un i suffisamment grand, montrant l'effet de la décomposition par paire, spécialement quand il y a suffisament de déterminisme. Une factorisation similaire est observée dans [Sánchez et al., 2004] pour les CSP.

Les figures 4.4 et 4.5 représentent l'état du graphe de contraintes avant et après le pré-traitement VE(i)+dec+bP, respectivement pour une instance des familles Grids et Linkage. Pour l'instance de la famille Grids sans le pré-traitement VE(3)+dec+bP seules deux variables auraient été éliminées.

Nous avons étudié le pré-traitement VE(i)+ dec+bP pour chaque instance des familles pour l'algorithme DFBB et également AND/OR-Branch-and-Bound (AOBB) avec cache et mini-bucket statique

14. graphmod.ics.uci.edu/uai08/Evaluation
15. www.cs.huji.ac.il/project/UAI10
16. http://graphmod.ics.uci.edu/uai08/Evaluation/Report/Benchmarks

	i=2	i=3	i=4	i=5	i=6	i=7
Linkage (22)						
DFBB-VE(i)	14	14	15	13	12	10
DFBB-VE(i)+dec	16	**19**	17	13	14	13
DFBB-VE(i)+bP	**17**	16	17	18	16	16
DFBB-VE(i)+dec+bP	16	18	**19**	**19**	**19**	**17**
Grids (32)						
DFBB-VE(i)	28	28	25	24	17	18
DFBB-VE(i)+dec	29	29	29	28	28	31
DFBB-VE(i)+bP	28	28	28	23	25	24
DFBB-VE(i)+dec+bP	**31**	**30**	**31**	**31**	**32**	**32**

Table 4.3 – Nombre d'instances résolues en moins d'une heure par chaque méthode

j-borné (AOBB-C+SMB(j)) qui a eu les meilleurs résultats pour Linkage et Grids à l'évaluation UAI'08. Pour utiliser au mieux AOBB-C+SMB(j) nous nous sommes restreints aux instances étudiées dans [Marinescu and Dechter, 2009] avec l'étude de la meilleure valeur de j.

La Table 4.4 donne les tailles des problèmes (n, d), la largeur d'arbre (w) et résume le temps CPU total utilisé le degré i choisi et j.

Pour DFBB et DFBB-VE(i)+dec+bP nous avons choisi la valeur de i qui permettait d'obtenir le meilleur résultat pour chaque instance ($i \in [2, 7]$). De manière générale, le degré d'élimination de variables est plus grand lorsqu'il est associé à la décomposition et au binProject. L'utilisation conjointe de ce degré plus grand et de dec+bP permet d'éliminer 10,5% de variables en plus pour les Linkages et 63,3% pour les Grids que DFBB-VE(i). Ce pourcentage est impressionnant pour la famille des Grids car pour DFBB-VE(i) les meilleurs résultats en temps ont été obtenus avec $i = 2$ ou 3 or la plupart des variables sont associées à plus d'une fonction ternaire, ce qui implique un degré supérieur à 3 (il est de 6 en général sur ces instances) donc peu de variables peuvent être éliminées. Mais ces instances possèdent beaucoup de fonctions décomposables (aidé par le déterminisme de certaines tables) ce qui permet de réduire les degrés de plusieurs variables et pouvoir les éliminer.

Après cette étude il peut paraître difficile de trouver une bonne valeur de i, mais en réalité un bon moyen de régler la valeur de i est de prendre la valeur correspondant au maximum de la distribution des degrés pour le modèle graphique considéré (ici, $i = 5$ pour Linkage et $i = 6$ pour Grids).

Pour AOBB, implémentée dans `aolibWCSP`[17]. La valeur de j est choisie comme dans [Marinescu and Dechter, 2009] et nous avons choisi pour i la valeur utilisée par DFBBVE(i)+dec+bP à chaque instance. Nous constatons que le pré-traitement dec+bP améliore également le temps de résolution de ces intances de Linkage et Grids. Les Grids qui n'étaient pas résolues avec AOBB-C+SMB(j)+VE(i), le sont en moins d'une seconde (en général) dès lors que nous utilisons notre pré-traitement.

17. graphmod.ics.uci.edu/group/aolibWCSP

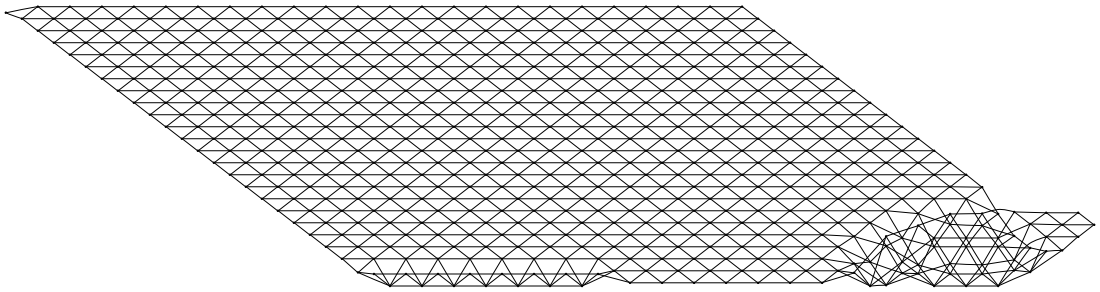
4.5 Conclusion

Ce chapitre nous avons permis de définir une décomposition par paire d'une fonction de coûts f illustrant dans le cas des modèles graphiques probabilistes une indépendance par paire dans le réseau associé à la distribution de Gibbs définie par f. Pour cette décomposition nous avons établi deux théorèmes, celui de la décomposabilité et celui de la construction des fonctions résultantes. Cette décomposition par paire combinée au *binProject* et à l'élimination de variable de degré borné est une puissante technique en pré-traitement de DFBB pour résoudre les problèmes MPE.

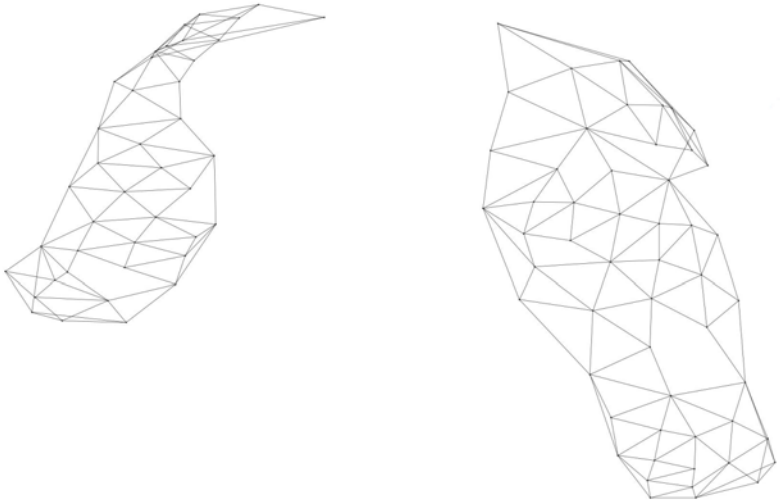
Nous avons présenté un début de travail sur la décomposition approchée qu'il faudra approfondir et expérimenter.

Problème				DFBB-VE(i)			DFBB-VE(i) +dec+bP			AOBB-C+SMB(j) +VE(i)				AOBB-C+SMB(j) +VE(i)+dec+bP	
	n	d	w	time (s)	i	n'	time (s)	i	n'	j	i	time (s)	n'	time (s)	n'
Linkage															
ped1	334	4	16	0.12	3	159	**0.07**	3	142	10	3	0.08	159	0.09	142
ped7	1068	4	38	4.04	2	375	**1.18**	4	213	20	4	-	289	131.40	213
ped9	1118	4	31	-			**3.36**	6	146	20	6	2225.40	197	94.20	146
ped18	1184	5	24	149.71	4	365	**3.19**	5	203	20	5	30.96	270	16.00	203
ped19	793	5	32	-			-			20	6	-	353	-	303
ped20	437	5	23	3.46	4	95	**0.39**	6	42	16	6	771.90	79	66.31	42
ped23	402	5	26	0.09	4	79	**0.05**	3	97	12	3	5.1	113	1.50	97
ped25	1289	5	29	1207.97	4	269	**0.65**	6	101	20	6	47.72	167	7.31	101
ped30	1289	5	24	543.20	2	633	**5.44**	5	222	20	5	20.42	282	3.34	222
ped33	798	4	30	0.84	3	272	**0.54**	6	95	18	6	925.12	193	4.95	95
ped34	1160	5	36	1.13	2	326	**0.36**	5	167	20	5	-	250	25.00	167
ped37	1032	4	22	0.21	5	103	**0.11**	4	108	10	4	264.92	153	138.07	108
ped38	724	5	18	0.24	5	122	**0.18**	5	96	12	5	261.62	135	105.97	96
ped39	1272	5	22	16.68	4	183	**0.24**	5	107	18	5	14.88	164	2.57	107
ped41	1062	5	35	-			**302.05**	4	296	20	4	-	375	1651.23	296
ped42	448	5	24	1.94	4	140	**0.34**	6	81	16	6	-	97	154.96	81
ped44	811	4	31	-			505.46	5	211	20	5	-	253	**83.74**	211
ped50	514	6	18	0.90	4	133	**0.18**	4	118	12	4	316.16	133	518.24	118
Grids															
90-24-1	576	2	35	0.07	3	529	**0.04**	3	96	18	3	78.20	529	0.01	96
90-26-1	676	2	41	0.29	3	632	**0.07**	3	303	16	3	3036.84	632	0.18	303
90-30-1	900	2	49	5.50	2	766	**0.26**	4	281	18	4	-	837	1.60	281
90-34-1	1156	2	63	328.42	2	1052	**0.48**	5	276	20	5	-	1086	0.15	276
90-38-1	1444	2	64	-			1.96	6	115	20	6	-	1036	**0.27**	115

Table 4.4 – Comparaison avec/sans le pré-traitement dec+bP pour DFBB-VE(i) et AOBB-C+SMB(j)

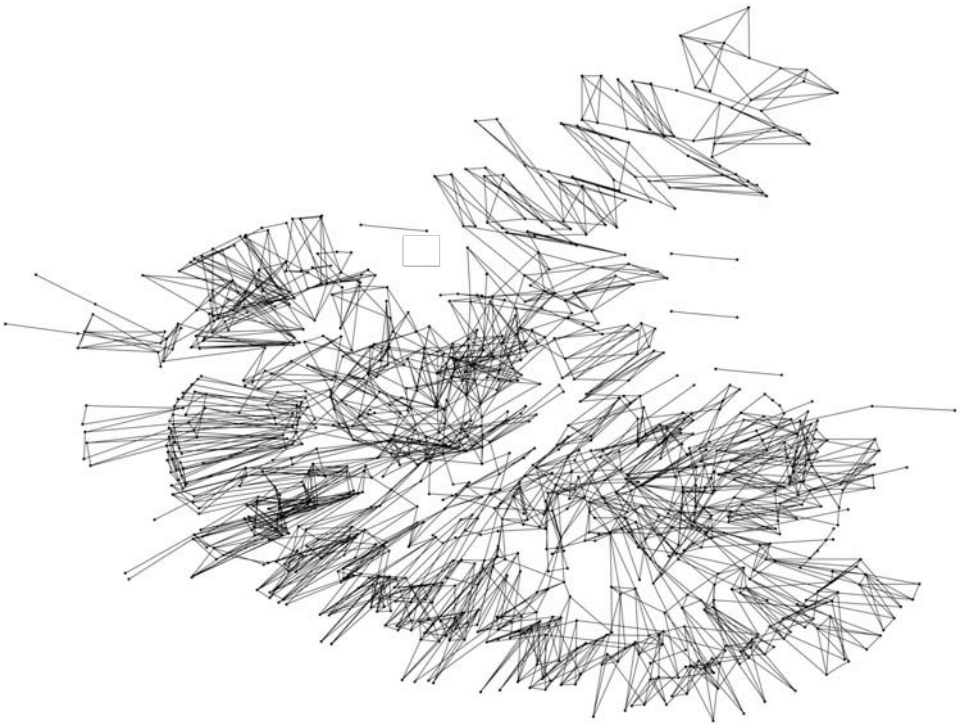


(a) Le graphe de contraintes original (576 sommets)

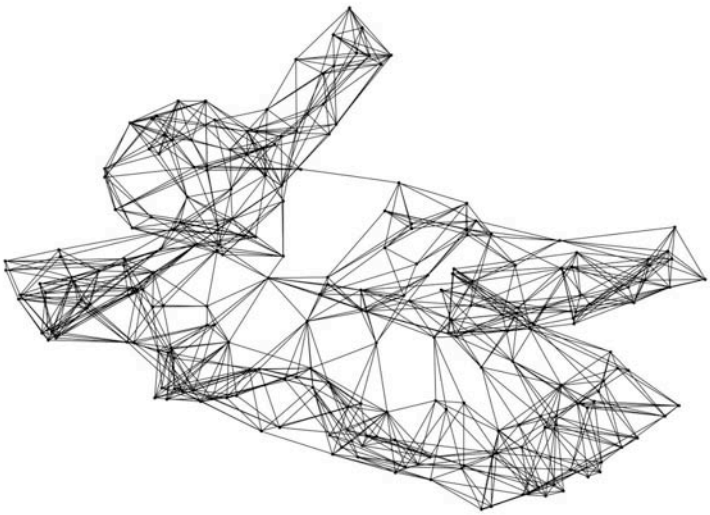


(b) Le graphe de contraintes résultant de `VE`(3)+dec+bP (96 sommets)

Figure 4.4 – Problème Grids *90-24-1* avant et après `VE`(3)+dec+bP



(a) Le graphe de contraintes original (1 289 sommets)



(b) Le graphe de contraintes résultant de `VE`(5)+dec+bP (222 sommets)

Figure 4.5 – Problème Linkage *pedigree30* avant et après `VE`(3)+dec+bP

Exploitation d'une décomposition structurelle pour le comptage de solutions 5

Ce chapitre présente une méthode de calcul exact pour le comptage dans le cadre des réseaux de contraintes basée sur une décomposition arborescente. La difficulté du comptage nous amène à présenter également une méthode de calcul approché en décomposant le problème en sous-problèmes pouvant être résolus par notre algorithme de calcul exact.

5.1 Etat de l'art

5.1.1 Comptage de solutions

Dans la littérature, deux principales approches ont été étudiées. D'un côté, les méthodes calculant le nombre exact de solutions et de l'autre, des méthodes calculant un nombre approché de solutions. Pour les méthodes de calcul exact, l'approche naturelle est d'étendre les méthodes systématiques telles que Forward-Checking (FC) [Haralick and Elliott, 1980] ou Maintaining Arc Consistency (MAC) [Sabin and Freuder, 1994] dans le but d'énumérer toutes les solutions. La complexité est bornée par $O(m.d^n)$ avec n le nombre de variables, m le nombre de contraintes, et d la taille maximum des domaines. Il est évident qu'avec cette approche, plus il y a de solutions, plus cela prend du temps pour les énumérer. Le problème de dénombrement de solutions est connu pour être #P-complet [Valiant, 1979].

Problème #Csp

Entrée : Une instance de problème Csp $\mathcal{P}$

Sortie : Le nombre de solutions de $\mathcal{P}$

Cependant, la plupart des méthodes de dénombrement ont été proposées dans le cadre de la logique propositionnelle (Sat, *Satisfiability*).

5.1.1.1 Formalisme SAT

Un problème SAT est un problème CSP $(\mathbf{X}, \mathbf{D}, \mathbf{C})$ dont toutes les variables sont booléennes de domaine $\{0, 1\}$ ou $\{faux, vrai\}$. A chaque variable X sont associés 2 littéraux, un positif (X) et un négatif $(\neg X)$. Les contraintes sont exprimées sous forme de disjonctions de littéraux appelées *clauses*. Chaque clause interdit un seul tuple dans la contrainte. Par exemple, $X \vee \neg Y \vee Z$ se lit *(X est vrai) ou (Y est faux) ou (Z est vrai)* et interdit le tuple (*faux, vrai, faux*). Comme dans les CSP le but est de satisfaire toutes les clauses, en logique on parle de la conjonction des clauses. Cette représentation est appelée CNF (*Conjonctive Normal Form*). On parle de *modèles* d'une formule propositionnelle plutôt que de solutions.

Problème SAT
Entrée : Une formule CNF F
Sortie : F admet-elle au moins un modèle ?

Exemple 5.1 *Soit la formule* CNF $F = \{(\neg A \vee \neg B \vee C) \wedge (A \vee \neg D \vee \neg E) \wedge (B \neg F)\}$. *Le problème* SAT $\mathcal{S}$ *associé est défini par* $\mathbf{X} = \{A, B, C, D, E, F\}$ *et* $\mathbf{C} = \{\neg A \vee \neg B \vee C, A \vee \neg D \vee \neg E, B \neg F\}$

$\mathcal{A} = \{A \leftarrow Faux, B \leftarrow Vrai, C \leftarrow Vrai, D \leftarrow Faux, E \leftarrow Faux, F \leftarrow Vrai\}$ *est un modèle de F (et de* $\mathcal{S}$*).*

De part la spécificité des contraintes (clauses) des instances SAT, il existe des traitements spéciaux pouvant être effectués, en particulier la détection des clauses satisfaites et la *propagation unitaire*. En effet, une clause est satisfaite dès lors qu'un littéral est satisfait [1].

La propagation unitaire supprime toutes les clauses unitaires (ne portant que sur un seul littéral) en affectant les variables associées pour satisfaire ces clauses. Ces clauses unitaires sont des contraintes unaires n'autorisant qu'une seule des deux valeurs v du domaine de la variable concernée, pour obtenir un modèle il faut donc forcément que cette variable soit affectée à la valeur v.

La propagation unitaire supprime ensuite toutes les clauses devenues satisfaites par ces affectations et tous les littéraux portant sur ces variables affectées dans les autres clauses (si $X \leftarrow vrai$, on supprime $\neg X$ des autres clauses).

Les algorithmes de base de la résolution du problème SAT sont similaires aux algorithmes des CSP. En règle générale, à partir d'une formule F nous choisissons une variable que nous affectons à $vrai$ ou à $faux$ et nous propageons ce choix. Si une clause est satisfaite elle est supprimée de la formule, dans le cas contraire le littéral associé à la variable affectée est supprimé (la clause est réduite). Dans le cas où la clause n'est pas satisfaite et devient vide, l'affectation courante n'admet pas d'extension vers un modèle. Lorsqu'une variable X est affectée à $vrai$ (resp. $faux$) nous pouvons appliquer la propagation unitaire sur $F \cap \{X\}$ (resp. $F \cap \{\neg X\}$) pour simplifier F.

1. le littéral X est satisfait si et seulement si $X \leftarrow vrai$ et le littéral $\neg X$ l'est si et seulement si $X \leftarrow faux$

Le problème de comptage de modèle associé au formalisme SAT est nommé #SAT.

Problème #SAT

Entrée : Une formule CNF F

Sortie : Le nombre de modèles de F

5.1.1.2 Calcul exact du nombre de solutions

Une première approche pour le dénombrement est basée sur l'extension des algorithmes de résolution comme Davis-Putnam, DPLL [Davis and Putnam, 1960] (équivalent à FC dans les CSP). Cette extension, nommée CDP (*Counting Davis-Putnam*), a été proposée dans [Birnbaum and Lozinskii, 1999], elle utilise la méthode DPLL pour explorer l'arbre de recherche. Lorsque toutes les clauses sont satisfaites (la formule est donc satisfaite), elle évalue le nombre k de variables déjà affectées et déduit le nombre de modèles pouvant être étendus à partir de cette affectation partielle : 2^{n-k}, où n est le nombre de variables de la formule. En effet, les $n-k$ variables non affectées n'ont plus de contraintes (clauses) portant sur elles, les deux valeurs de leur domaine peuvent être utilisées pour atteindre un modèle. Ce calcul est ajouté à ceux déjà obtenus au cours de la recherche. La méthode est décrite par l'algorithme 4 dont l'appel initial est CDP$(F, \emptyset, 0)$.

Algorithme 4: CDP (F,$\mathcal{A}$, k) : entier

Entrée :
une formule CNF F
une affectation $\mathcal{A}$
un entier k

Pré-conditions :
k est le nombre de variables affectées dans $\mathcal{A}$, *i.e.* , $k =\mid \mathcal{A} \mid$

Post-relations :
renvoie le nombre de modèles ayant comme affectation partielle $\mathcal{A}$

$k \leftarrow$ `PropagationUnitaire(F,k)`;
if *Il existe une clause vide dans F* **then return** 0;
if *Toutes les clauses de F sont satisfaites* **then**
  return 2^{n-k};
Choisir X une variable non affectée de F;
return *CDP(*$F, \mathcal{A} \cup \{X \leftarrow vrai\}, k+1$*)+ CDP(*$F, \mathcal{A} \cup \{X \leftarrow faux\}, k+1$*);*

Dans le même esprit l'algorithme DDP (*Decomposing Davis-Putnam*) [Bayardo and Pehoushek, 2000] se base sur l'algorithme DPLL. Par rapport à CDP, l'algorithme ajoute une étape d'identification des composantes connexes du graphe de contraintes de la formule F, représentant des sous-formules disjointes après la propagation unitaire de la formule. Chaque sous-formule peut être traitée indépendamment, pour déterminer son nombre de solutions avec ce même processus. Cet algorithme est implémenté dans le logiciel `Relsat`.

Par la propriété 5.2, suivante, le nombre de solutions de F est le produit de celui de ses sous-formules identifiées par les composantes connexes du graphe de contraintes.

Propriété 5.2 *Soit k problèmes $\mathcal{P}_1, \ldots, \mathcal{P}_k$ disjoints (c'est-à-dire $\mathbf{X}_i \cap \mathbf{X}_j = \emptyset$, $\forall i, j$), et $\mathcal{S}_{\mathcal{P}_i}$ le nombre de solutions du problème $\mathcal{P}_i$. Si $\mathcal{P} = \bigcup_{i=1}^{k} \mathcal{P}_i$ alors $\mathcal{S}_{\mathcal{P}} = \prod_{i=1}^{k} \mathcal{S}_{\mathcal{P}_i}$.*

La décomposition effectuée à chaque affectation de variables permet d'éviter certains calculs redondants par rapport à CDP du fait des ensembles de variables disjoints. Mais d'autres calculs sont encore dupliqués lorsque nous nous retrouvons avec une même sous-formule rencontrée plus tôt dans la recherche. Au lieu de refaire ces calculs, `Cachet` [Sang et al., 2004] enregistre le résultat des calculs déjà réalisés en amont dans la recherche. Cet enregistrement d'informations permet en théorie à `Cachet` d'être plus performant que DDP.

Une approche différente consiste à transformer (compiler) la formule CNF en une autre forme logique dans laquelle le comptage serait plus facile (polynomial par rapport à la taille de la nouvelle forme logique). Darwiche dans [Darwiche, 2004] présente un compilateur de connaissances `c2d` qui convertit une formule CNF en une formule NNF (negative normal form) déterministe, et décomposable d-DNNF [Darwiche, 2001]. Une formule NNF est une disjonction de conjonctions de littéraux [2].

Une formule NNF peut se représenter à l'aide d'un graphe orienté acyclique, tel que l'étiquette de chaque puits est un littéral, et les étiquettes des autres sommets sont les opérateurs AND et OR. La figure 5.1 représente le graphe d'une formule NNF. La formule est *décomposable* lorsque pour chaque enfant d'un sommet AND les variables apparaissant dans sa descendance sont disjointes de celle de ses frères. La formule est *déterministe* lorsque les formules induites par un sommet OR ne peuvent pas être satisfaites en même temps [3]. Le comptage des solutions à partir d'un sommet AND revient à multiplier le nombre de solutions des sous-formules induites par la descendance de ses enfants (application directe de la propriété 5.2). Le nombre de solutions à partir d'un sommet OR s'obtient par la somme des sous-formules induites par la descendance de ses enfants.

A partir de cette représentation, pour obtenir le nombre de solutions de la formule F, il suffit de parcourir le graphe en partant des puits et remonter vers la source. A chaque sommet du graphe va lui être associé le nombre de solutions de la formule induite par le sommet et sa descendance. Pour les puits (représentant les littéraux) leur nombre de solutions est trivialement 1. Au final, le nombre de solutions de la formule F est contenu dans la source du graphe associé.

Soit la formule $F = \{(\neg A \vee \neg B \vee \neg C \vee \neg D) \wedge (\neg A \vee \neg B \vee C \vee D) \wedge (\neg A \vee B \vee \neg C \vee D) \wedge (\neg A \vee B \vee C \vee \neg D) \wedge (A \vee \neg B \vee \neg C \vee D) \wedge (A \vee \neg B \vee C \vee \neg D) \wedge (A \vee B \vee \neg C \vee \neg D) \wedge (A \vee B \vee C \vee D)\}$. La formule est représentée par la figure 5.1, elle admet 8 solutions. `c2d` se montre très compétitif et parfois plus efficace que les méthodes basé sur DPLL comme `Relsat` ou `Cachet`. Le compilateur `c2d` à

2. contrairement à une formule CNF qui est une conjonction de disjonction de littéraux.
3. par exemple pour la formule $(\neg A \wedge B) \vee (A \wedge \neg B)$, les sous-formules $(\neg A \wedge B)$ et $(A \wedge \neg B)$ ne peuvent pas être satisfaites ensemble car $(\neg A \wedge B) \wedge (A \wedge \neg B)$ n'admet pas de modèle.

partir de la méthode de comptage de solution peut également calculer la probabilité marginale d'une probabilité.

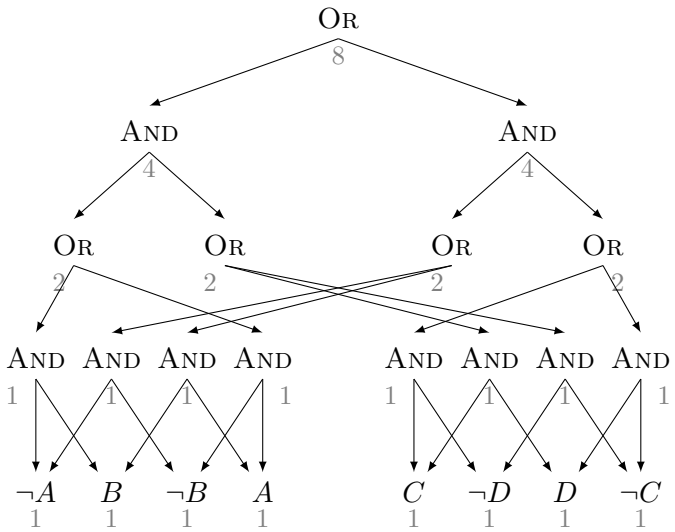


Figure 5.1 – Le graphe représentant une formule NNF (issu de [Darwiche, 2004]). En gris est indiquée la valeur des étiquettes, la formule admet 8 solutions.

Dans le cadre des CSP, quelques méthodes existent. Dechter et Mateescu ont développé un algorithme, appelé AND/OR-Counting [Dechter and Mateescu, 2002], utilisant un arbre de recherche de type AND/OR pour compter le nombre de solutions des CSP. Les auteurs proposent de détecter les indépendances entre les sous-problèmes à l'aide d'un pseudo-arbre.

Définition 5.3 Pseudo-arbre [Freuder and Quinn, 1985]
Un pseudo-arbre $T = (\mathbf{X}, \mathbf{C}')$ de $G = (\mathbf{X}, \mathbf{C})$ est une arborescence telle que toute arête de $\mathbf{C}$ n'appartenant pas à $\mathbf{C}'$ relie un sommet de $\mathbf{X}$ à l'un de ses ancêtres dans T.

La figure 5.2b est un pseudo-arbre associé au graphe 5.2a.

A partir d'un sommet de ce pseudo-arbre, chaque sous-arbre enraciné par ses enfants est indépendant des autres (l'ensemble de ses variables est disjoint des autres), la propriété 5.2 permet de déduire qu'il suffira d'effectuer le produit du nombre de solutions de ces sous-problèmes, pour obtenir le nombre de solutions de leur union.

Pour prendre en compte ces indépendances Dechter et Mateescu utilisent un arbre de recherche AND/OR.

Définition 5.4 Arbre de recherche AND/OR
Soient le CSP $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{C})$ et $T = (\mathbf{X}, \mathbf{C}')$ un pseudo-arbre du graphe de contraintes $G = (\mathbf{X}, \mathbf{C})$. L'arbre de recherche AND/OR admet des niveaux alternés de sommets de type AND et de type OR [4].

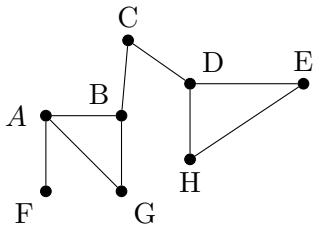
4. Un sommet de type OR n'admet comme fils que des sommets de type AND et vice versa.

Les sommets Or représentent les variables X_i et chaque sommet And, noté $\langle X_i, x_i \rangle$, représente la valeur x_i affectée à la variable X_i (associée au sommet Or père).

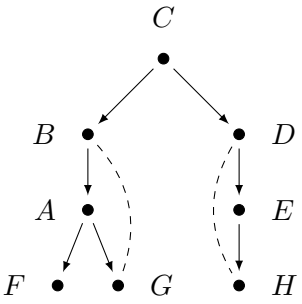
- Un sommet Or X_i admet comme enfant un sommet And $\langle X_i, x_i \rangle$ si et seulement si l'affectation de X_i à x_i est cohérente avec l'affectation partielle définie par le chemin de la racine au sommet Or X_i.
- Un sommet And $\langle X_i, x_i \rangle$ admet comme enfant un sommet Or X_j si et seulement si X_j est un enfant de X_i dans le pseudo-arbre T.

La figure 5.2c est un extrait de l'arbre de recherche And/Or du Csp défini à la figure 5.2.

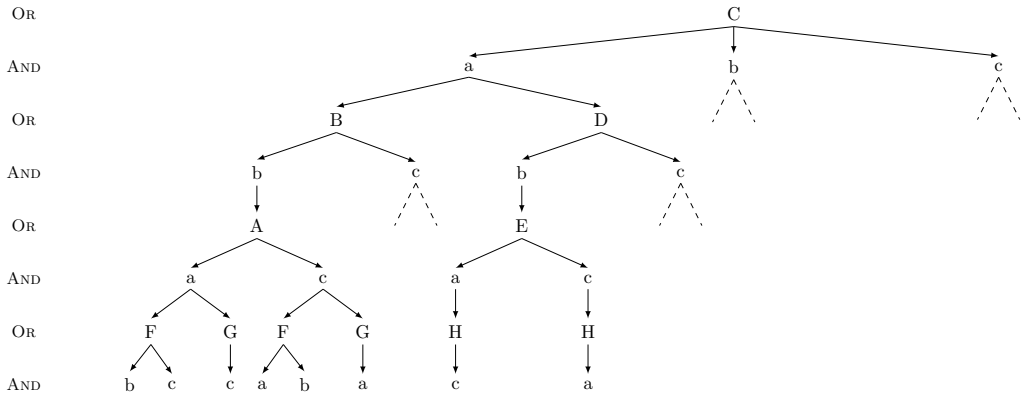
Pour obtenir le nombre de solutions d'un Csp $\mathcal{P}$, le principe est similaire à `c2d`, seul diffère l'étiquetage des puits (ici les feuilles). Si une feuille est un sommet de type Or l'étiquette associée au sommet est 0 (l'affectation n'est pas cohérente), dans le cas d'un sommet feuille de type And l'étiquette associée est 1. Il suffit ensuite de faire comme `c2d`, nous multiplions les étiquettes des enfants des sommets And et nous additionnons les étiquettes des enfants des sommets Or. L'étiquette de la racine est alors le nombre de solutions du problème $\mathcal{P}$.



(a) Graphe de contraintes



(b) Pseudo-arbre



(c) Une partie de l'arbre de recherche And/Or

Figure 5.2 – Les structures pour And/Or-counting pour une instance Csp de 7 variables de domaine $\{a, b, c\}$ avec des contraintes binaires de différence.

Cet algorithme admet une complexité linéaire en espace et en $O(n \exp(w^* \log(n))$ en temps où w^* est la largeur d'arbre du problème.

Cependant, pour éviter certaines redondances, [Dechter and Mateescu, 2002] propose une modification de l'algorithme en utilisant la notion de séparateur-parent pour chaque sommet défini par l'ensemble constitué de ses ancêtres dans T et lui-même. Ce qui nous permet d'obtenir un graphe (minimal) de recherche And/Or. Cette modification entraînant la construction de ce graphe est exponentielle à la largeur d'arbre du problème.

Comparée à une méthode de type Backtrack adaptée au comptage, cette méthode est bien meilleure en terme de nœuds et de temps d'exécution, surtout lorsque le nombre de solutions est important. Nous voyons encore ici l'intérêt qu'il faut porter à la décomposition du problème.

5.1.1.3 Calcul approché du nombre de solutions

La difficulté que rencontrent les algorithmes de calcul exact du nombre de solutions est le parcours de l'espace de recherche, même si celui-ci peut être filtré ou décomposé. Une alternative est de n'en parcourir qu'une partie et de se contenter d'une approximation du nombre de solutions. Différentes manières d'établir une approximation de ce nombre de solutions ont été explorées, de la simple approximation, avec et sans garanties, à l'établissement d'une borne inférieure et/ou supérieure.

De la même manière que pour le calcul exact, la recherche sur ces méthodes s'est faite principalement pour les instances Sat.

Wei et Selman [Wei and Selman, 2005] proposent une recherche locale basée sur de l'échantillonnage MCMC (Markov Chain Monte Carlo) pour obtenir un calcul approché sans garantie. Leur algorithme `ApproxCount`, se base sur l'idée suivante : *la répartition des affectations partielles des solutions sur un sous-ensemble est la même que sur l'ensemble complet de solutions.* Ainsi, à partir d'un échantillon de k solutions du problème, il évalue les proportions de solutions contenant une affectation partielle particulière et étend cette proportion à l'ensemble complet des solutions. Par exemple, sur un échantillon de k modèles d'une formule F, l'affectation $X_i \leftarrow vrai$ correspond à k^+ modèles (sur les k modèles), $\gamma = \frac{k^+}{k}$ est le ratio des modèles dans l'échantillon ayant X_i affectée à $vrai$. Immédiatement nous avons $k = \frac{1}{\gamma}k^+$, avec (l'hypothèse que $\gamma > 0$). La formule est alors simplifiée par la propagation unitaire de $X_i \leftarrow vrai$, il suffit de recommencer en calculant un ensemble de modèles de cette nouvelle formule, jusqu'à ce que la formule restante soit vide ou qu'elle soit suffisamment facile pour pouvoir compter exactement son nombre de modèles. `ApproxCount` évalue finalement le nombre de modèles de la formule F par le produit des $\frac{1}{\gamma}$ et le nombre exact de modèles de la dernière formule. Pour s'assurer que γ soit le plus possible strictement supérieur à zéro `ApproxCount` affecte chaque variable à la valeur dont le nombre de modèles dans l'échantillon considéré est le plus grand.

Sa complexité est majorée par $(n \cdot k \cdot t)$ où k est la taille de l'échantillon, et t le temps nécessaire à la récupération d'une solution, ce dernier est exponentiel à la taille de la formule. Par rapport aux

méthodes de calcul exact `ApproxCount` est très rapide et, la plupart du temps, propose une bonne estimation du nombre de modèles.

Dans [Lerman and Rouat, 1999] les auteurs utilisent le principe de "*diviser pour résoudre*" pour reprendre leur expression. Leur méthode consiste à diviser une formule de départ en deux formules de taille similaire, en terme de nombre de variables, et les plus indépendantes possibles. Le nombre de modèles de chacune des deux formules est établi de manière exacte, le nombre de modèles de la formule de départ est alors approximé par le produit du nombre de modèles des deux formules créées et $\frac{1}{2^n}$. La qualité de l'approximation dépend de l'indépendance réelle entre les deux formules créées.

Gomes et al. proposent dans [Gomes et al., 2007] une variante de `ApproxCount`, `SampleCount`, permettant d'obtenir une borne inférieure du nombre de solutions, plutôt qu'une approximation, avec une garantie probabiliste. La principale différence entre les deux méthodes réside dans le choix des variables à affecter. `SampleCount` choisit une variable dont le nombre de modèles dans l'échantillon est quasiment le même qu'elle soit affectée à $vrai$ ou à $faux$, il va choisir la variable qui s'en rapproche le plus. Pour obtenir une borne inférieure du nombre de solutions d'une formule, `SampleCount` va itérer le processus et garder le calcul minimum parmi toutes les itérations. Il est prouvé dans [Gomes et al., 2007] que la borne obtenue par `SampleCount` est correcte avec une probabilité de $1 - 2^{-\alpha t}$ où t est le nombre d'itérations et α une valeur, ils font partie des paramètres d'entrée de `SampleCount` avec également k la taille des échantillons. Lorsque $\alpha t = 7$ cette probabilité est de 99%. Comme pour `ApproxCount` à chaque affectation de variables un échantillon de k modèles doit être construit. Même si comparée à `ApproxCount` la méthode est moins rapide du fait des t itérations, elle donne une borne et reste plus compétitive que les méthodes de calcul exact.

Même si elle n'a été testée que sur des instances Sat, la méthode proposée par [Bailleux and Chabrier, 1996] peut s'appliquer aux Csp. Elle évalue le nombre de solutions en estimant l'espérance d'une variable aléatoire, définie sur un ensemble de chemins de l'arbre de recherche partant de la racine vers ses feuilles avec pour base l'algorithme Forward-Checking. Elle permet d'obtenir une borne inférieure, mais aucune garantie quant à une borne supérieure.

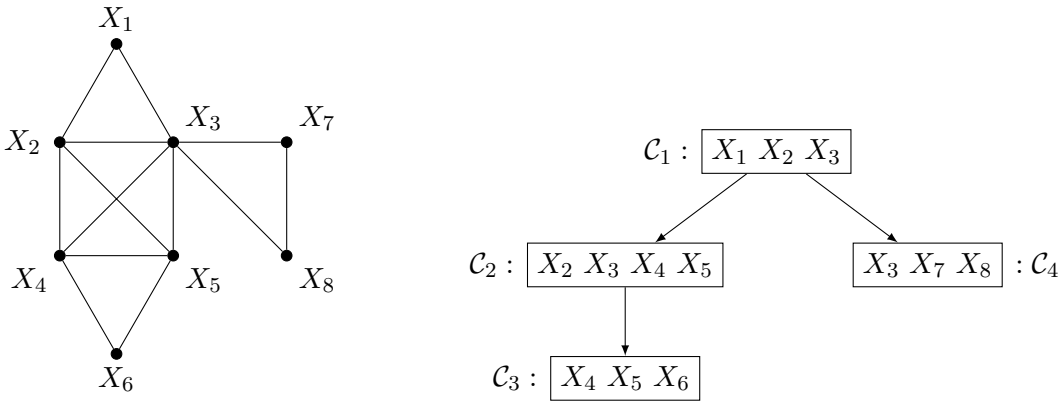
5.1.2 L'algorithme Btd

Les méthodes structurelles sont efficaces pour la résolution des Csp. L'une d'entre elles est l'algorithme Btd (*Backtrack Tree Decomposition*) [Jégou and Terrioux, 2003]. Dans la section suivante nous l'adaptons au comptage de solutions.

La première étape de Btd consiste à calculer une décomposition arborescente du graphe de contraintes. La figure 5.3 présente le graphe de contraintes d'un problème $\mathcal{P}$, décrit dans la légende, et une décomposition arborescente de celui-ci.

Cette décomposition arborescente permet d'obtenir un ordre partiel sur les variables permettant à BTD d'exploiter les propriétés structurelles du graphe et de couper dans l'arbre de recherche. En effet, les variables sont affectées selon une recherche en profondeur d'abord à partir de la racine de la décomposition. Autrement dit, nous affectons les variables du cluster racine $\mathcal{C}_1$, puis celles de $\mathcal{C}_2$, celles de $\mathcal{C}_3$ etc. Par exemple, $X_1, X_2, \ldots, X_8$ est un ordre possible.

De plus, la décomposition arborescente et l'ordre sur les variables permettent à BTD de diviser le problème $\mathcal{P}$ en plusieurs sous-problèmes. Etant donnés deux clusters $\mathcal{C}_i$ et $\mathcal{C}_j$ (avec $\mathcal{C}_j$ un fils de $\mathcal{C}_i$), le sous-problème enraciné en $\mathcal{C}_j$ dépend de l'affectation courante $\mathcal{A}$ sur le séparateur $\mathcal{C}_i \cap \mathcal{C}_j$, on notera ce sous-problème $\mathcal{P}_{\mathcal{A},\mathcal{C}_i/\mathcal{C}_j}$. Son ensemble de variables est $Desc(\mathcal{C}_j)$. Le domaine de chaque variable appartenant à $\mathcal{C}_i \cap \mathcal{C}_j$ est réduit à sa valeur associée dans $\mathcal{A}$. Concernant l'ensemble des contraintes, il contient les contraintes qui impliquent au moins une variable apparaissant exclusivement dans $\mathcal{C}_j$ ou un de ses descendants.



(a) Un graphe de contraintes de $\mathcal{P}$ **(b)** Une décomposition arborescente pour $\mathcal{P}$

Figure 5.3 – Le CSP $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{C})$ tel que chaque domaine des variables est $\{a, b, c, d\}$ et chaque contrainte $c_{ij} = \{X_i, X_j\}$ est une contrainte de différence : $X_i \neq X_j$.

Exemple 5.5 *Considérons le* CSP $\mathcal{P}$ *de la figure 5.3. Soit* $\mathcal{A} = (X_2 \leftarrow b, X_3 \leftarrow c)$*, l'ensemble de variables de* $\mathcal{P}_{\mathcal{A},\mathcal{C}_1/\mathcal{C}_2}$ *est* $Desc(\mathcal{C}_2) = \{X_2, X_3, X_4, X_5, X_6\}$*, (avec* $d_{X_2} = \{b\}, d_{X_3} = \{c\}$ *et* $d_{X_4} = d_{X_5} = d_{X_6} = \{a, b, c, d\}$*) et son ensemble de contraintes est* $\{c_{24}, c_{25}, c_{34}, c_{35}, c_{45}, c_{46}, c_{56}\}$.

La résolution du problème CSP est facilitée par l'enregistrement de *(no)goods* durant la recherche.

Définition 5.6 good et nogood

Un *good structurel* de $\mathcal{C}_i$ par rapport à $\mathcal{C}_j$ est l'affectation courante $\mathcal{A}$ sur $\mathcal{C}_i \cap \mathcal{C}_j$ pouvant être étendue sur le sous-problème $\mathcal{P}_{\mathcal{A},\mathcal{C}_i/\mathcal{C}_j}$.

Un *nogood structurel* de $\mathcal{C}_i$ par rapport à $\mathcal{C}_j$ est l'affectation courante $\mathcal{A}$ sur $\mathcal{C}_i \cap \mathcal{C}_j$ ne pouvant pas être étendue sur le sous-problème $\mathcal{P}_{\mathcal{A},\mathcal{C}_i/\mathcal{C}_j}$.

Exemple 5.7 *Considérons le* Csp $\mathcal{P}$ *de la figure 5.3 et l'affectation* $\mathcal{A} = \{X_2 \leftarrow b, X_3 \leftarrow c\}$ *sur* $\mathcal{C}_1 \cap \mathcal{C}_2$ *nous obtenons le good* $(\mathcal{A})$ *grâce à l'affectation sur* $Desc(\mathcal{C}_2)$ *défini par* $\{X_2 \leftarrow b, X_3 \leftarrow c, X_4 \leftarrow a, X_5 \leftarrow d, X_6 \leftarrow b\}$ *satisfaisant toutes les contraintes de* $\mathcal{P}_{\mathcal{A},\mathcal{C}_1/\mathcal{C}_2}$. *Dans un premier temps cette affectation* $(\mathcal{A})$ *est explorée, puis lorsque son extension sur* $\{X_4, X_5, X_6\}$ *est valide,* $(\mathcal{A})$ *est enregistrée comme un good. Ainsi, si durant la recherche, l'affectation* $\mathcal{A} = \{X_2 \leftarrow b, X_3 \leftarrow c\})$ *est à nouveau étudiée, la recherche n'explore pas le sous-problème* $\mathcal{P}_{\mathcal{A},\mathcal{C}_1/\mathcal{C}_2}$ *car nous avons déjà prouvé qu'il existe une solution compatible avec* $\mathcal{A}$. *Au contraire, si aucune solution n'est trouvée pour une autre affectation* $\mathcal{A}'$ *sur* $\mathcal{P}_{\mathcal{A}',\mathcal{C}_1/\mathcal{C}_2}$, *un* nogood structurel *est alors enregistré.*

Remarque 5.8 *Les nogoods pourront être utilisés comme de nouvelles contraintes du problème.*

Une telle mémorisation permet alors à Btd d'élaguer l'arbre de recherche en essayant de ne pas affecter une descendance d'un cluster alors que l'on sait déjà que cela sera (resp. ne sera pas) possible. En contre partie, l'espace mémoire requis est souvent important. En effet, le nombre de (no)goods pouvant être enregistré est exponentiel par rapport à la taille du plus grand séparateur de la décomposition arborescente. La complexité en espace de Btd est en $O(nsd^s)$, où n est le nombre de clusters, d la taille du plus grand domaine et s la taille du plus grand séparateur. Sa complexité temporelle est en $O(nmd^{w+1})$ où m est le nombre de contraintes et w la largeur de la décomposition arborescente traitée par Btd.

5.2 Une nouvelle méthode pour le comptage

Les méthodes structurelles comme Btd sont efficaces pour la résolution de Csp structurés. Nous proposons d'adapter l'algorithme Btd au dénombrement exact de solutions et ainsi évaluer l'intérêt de l'utilisation de la structure d'une instance pour le comptage de solutions : #Btd.

Comme pour Btd, la première étape de #Btd consiste à calculer une décomposition arborescente du graphe de contraintes. Tandis que Btd utilise la notion de good, pour le dénombrement de solutions nous utilisons la notion de *#good*. Le but est de sauver le nombre de solutions des sous-problèmes induits par la décomposition arborescente.

Définition 5.9 #good
Soit une décomposition arborescente $\mathcal{D}$. Étant donnés le cluster $\mathcal{C}_i$ et un de ses fils $\mathcal{C}_j$, un *#good* est une paire $(\mathcal{A}[\mathcal{C}_i \cap \mathcal{C}_j], nb)$ avec $\mathcal{A}[\mathcal{C}_i \cap \mathcal{C}_j]$ une affectation consistante sur $\mathcal{C}_i \cap \mathcal{C}_j$ et nb est le nombre de solutions du sous-problème $\mathcal{P}_{\mathcal{A},\mathcal{C}_i/\mathcal{C}_j}$.

Reprenons le Csp de la figure 5.3 page précédente. Si nous considérons l'affectation $\mathcal{A} = \{X_4 \leftarrow a, X_5 \leftarrow d\}$ sur $\mathcal{C}_2 \cap \mathcal{C}_3$ nous obtenons un #good $(\mathcal{A}, 2)$ car nous avons deux solutions pour le sous-problème $\mathcal{P}_{\mathcal{A},\mathcal{C}_2/\mathcal{C}_3}$: $\mathcal{A}_1 = \{X_4 \leftarrow a, X_5 \leftarrow d, X_6 \leftarrow b\}$ et $\mathcal{A}_2 = \{X_4 \leftarrow a, X_5 \leftarrow d, X_6 \leftarrow c\}$

Algorithme 5: #BTD($\mathcal{A}$, $\mathcal{C}_i$, $\mathbf{X}_{\mathcal{C}_i}$) : entier

Entrée :
une affectation $\mathcal{A}$
un cluster $\mathcal{C}_i$
un ensemble de variables $\mathbf{X}_{\mathcal{C}_i} \subseteq \mathbf{X}$

Pré-conditions :
$\mathbf{X}_{\mathcal{C}_i}$ contient les variables de $\mathcal{C}_i$ qui ne sont pas affectés dans $\mathcal{A}$

Post-relations :
renvoie le nombre de solutions du problème enraciné en $\mathcal{C}_i$ ayant comme affectation partielle $\mathcal{A}$

---

if $\mathbf{X}_{\mathcal{C}_i} = \emptyset$ **then**
  $NbSol \leftarrow 1$;
  $F \leftarrow$ `Fils`$(\mathcal{C}_i)$;
1 **while** $F \neq \emptyset$ ***et*** $NbSol \neq 0$ **do**
    Choisir $\mathcal{C}_j$ dans F;
    $F \leftarrow F \setminus \{\mathcal{C}_j\}$;
    if $(\mathcal{A}[\mathcal{C}_i \cap \mathcal{C}_j], nb)$ *n'est pas un #good dans* $\mathcal{P}$ **then**
      $nb \leftarrow$ #BTD$(\mathcal{A}, \mathcal{C}_j, \mathbf{X}_{\mathcal{C}_j} \setminus (\mathcal{C}_i \cap \mathcal{C}_j))$;
      enregistre le #good $(\mathcal{A}[\mathcal{C}_i \cap \mathcal{C}_j], nb)$ de $\mathcal{C}_i/\mathcal{C}_j$ dans $\mathcal{P}$;
    $NbSol \leftarrow NbSol \times nb$;
2 **return** $NbSol$;
else
  Choisir $X \in \mathbf{X}_{\mathcal{C}_i}$;
  $NbSol \leftarrow 0$;
  $\mathbf{D} \leftarrow D_X$;
3 **while** $D \neq \emptyset$ **do**
    Choisir v dans $\mathbf{D}$;
    $\mathbf{D} \leftarrow \mathbf{D} \setminus \{v\}$;
4 **if** $\mathcal{A} \cup \{X \leftarrow v\}$ *ne viole aucune contrainte* $c \in C$ **then**
      $NbSol \leftarrow NbSol +$ #BTD$(\mathcal{A} \cup \{X \leftarrow v\}, \mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i} \setminus \{X\})$;
5 **return** $NbSol$;

5.2.1 L'algorithme #BTD

#BTD explore l'espace de recherche suivant l'ordre des variables induit par la décomposition arborescente. Ainsi, on commence par les variables du cluster racine $\mathcal{C}_1$. A l'intérieur du cluster $\mathcal{C}_i$, on affecte une valeur à une variable, on "backtrack" si une contrainte est violée. Ce schéma peut être amélioré avec le maintien de l'arc consistance (ligne 4). Lorsque toutes les variables de $\mathcal{C}_i$ sont affectées, #BTD calcule le nombre de solutions du sous-problème induit par le premier fils de $\mathcal{C}_i$, s'il en existe un. Plus généralement, considérons $\mathcal{C}_j$ un fils de $\mathcal{C}_i$. Étant donnée une affectation courante $\mathcal{A}$ sur $\mathcal{C}_i$, #BTD vérifie si l'affectation $\mathcal{A}[\mathcal{C}_i \cap \mathcal{C}_j]$ correspond à un #good. Si c'est le cas, #BTD multiplie le nombre de solutions enregistré avec le nombre de solutions de $\mathcal{C}_i$ avec $\mathcal{A}$ comme affectation. Sinon, on étend $\mathcal{A}$ sur $Desc(\mathcal{C}_i)$ dans l'ordre pour compter le nombre nb d'extensions consistantes et on enregistre le #good $(\mathcal{A}[\mathcal{C}_i \cap \mathcal{C}_j], nb)$. #BTD calcule le nombre de solutions du sous-problème induit par le fils suivant de $\mathcal{C}_i$ (ligne 1). Finalement lorsque chaque fils de $\mathcal{C}_i$ a été examiné, #BTD backtrack et essaye de modifier l'affectation courante de $\mathcal{C}_i$. Le nombre de solutions de $\mathcal{C}_i$ est la somme des solutions de chaque affectation.

#BTD est décrit par l'algorithme 5. Le premier appel est BTD$(\emptyset, \mathcal{C}_1, \mathcal{C}_1)$ et il renvoie le nombre de solutions.

Exemple 5.10 *Reprenons le* CSP *de la figure 5.3 page 97. Soit l'affectation* $\mathcal{A} = \{X_1 \leftarrow a, X_2 \leftarrow b, X_3 \leftarrow c\}$ *comme affectation de* $\mathcal{C}_1$ *alors* $\mathcal{S}_{\mathcal{A},\mathcal{C}_1/\mathcal{C}_2} = \mathcal{S}_{\mathcal{A}\cup\{X_4\leftarrow a,X_5\leftarrow d\},\mathcal{C}_1/\mathcal{C}_2} + \mathcal{S}_{\mathcal{A}\cup\{X_4\leftarrow d,X_5\leftarrow a\},\mathcal{C}_1/\mathcal{C}_2} = 2 + 2 = 4$.

Le nombre de solutions de $\mathcal{P}_{\mathcal{A},\mathcal{C}_1/\mathcal{C}_2}$ *est* 4 *sur* $\{X_4, X_5, X_6\}$ *et* 6 *solutions pour* $\mathcal{P}_{\mathcal{A},\mathcal{C}_1/\mathcal{C}_4}$ *sur* $\{X_7, X_8\}$. *Donc, il y a* 24 *solutions sur* $\{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8\}$ *avec* $\mathcal{A}$.

Notons que pour $[\mathcal{C}_1 \cap \mathcal{C}_4]$, $(\{X_3 \leftarrow c\}, 6)$ *est un #good. Pour une autre affectation sur* $\mathcal{C}_1$, *par exemple* $\mathcal{A}' = \{X_1 \leftarrow b, X_2 \leftarrow a, X_3 \leftarrow c\}$, *il n'est pas nécessaire de calculer les solutions sur* $\mathcal{C}_4$ *car le #good* $(\{X_3 \leftarrow c\}, 6)$ *peut être utilisé pour* $\mathcal{A}'$.

5.2.2 Propriétés

Théorème 5.11 *L'algorithme* #BTD *est correct, complet et termine.*

Démonstration. Cette preuve s'effectue par induction en exploitant les propriétés des #goods structurels. L'induction est réalisée sur le nombre de variables apparaissant dans la descendance du cluster $\mathcal{C}_i$, exceptée celle de $\mathcal{C}_i$ déjà affectées. Cet ensemble de variables est noté :

$$\mathcal{V}ar_i = \mathcal{V}ar(\mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i}) = \mathbf{X}_{\mathcal{C}_i} \cup \left(\bigcup_{\mathcal{C}_j \in \texttt{fils}(\mathcal{C}_i)} Desc(\mathcal{C}_j) \setminus (\mathcal{C}_i \cap \mathcal{C}_j) \right)$$

$\mathcal{V}ar(\mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i})$ correspond à l'ensemble des variables à affecter pour déterminer le nombre d'extensions cohérentes de $\mathcal{A}$ sur les variables $\mathbf{X}_{\mathcal{C}_i}$ et sa descendance $(Desc(\mathcal{C}_i))$.

Pour démonter que l'algorithme #BTD est correct, nous devons prouver la propriété $P(\mathcal{A}, \mathcal{V}ar(\mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i}))$ définie par : "#BTD$(\mathcal{A}, \mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i})$ *renvoie le nombre d'extensions cohérentes de* $\mathcal{A}$ *sur les variables* $\mathbf{X}_{\mathcal{C}_i}$ *et sur la descendance de* $\mathcal{C}_i$ *".*

Initialisation : Considérons $P(\mathcal{A}, \emptyset)$

Si $\mathcal{V}ar(\mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i}) = \emptyset$ alors $\mathbf{X}_{\mathcal{C}_i} = \emptyset$ et $\texttt{fils}(\mathcal{C}_i) = \emptyset$. Puisque $\mathcal{A}$ est une affectation cohérente sur $\mathbf{X}_{\mathcal{C}_i}$ et sur la descendance de $\mathcal{C}_i$. Il n'y a qu'une seule extension de $\mathcal{A}$ sur $\mathbf{X}_{\mathcal{C}_i}$ et sur la descendance de $\mathcal{C}_i$, $\mathcal{A}$ elle-même. Donc $P(\mathcal{A}, \emptyset)$ est vérifiée.

Induction : $P(\mathcal{A}, \mathcal{V}ar_i)$ avec $\mathcal{V}ar_i \neq \emptyset$. Supposons que $\forall \mathcal{V}ar'_i \subset \mathcal{V}ar_i$ $P(\mathcal{A}, \mathcal{V}ar'_i)$ soit vérifiée.

1^{er} **cas :** $\mathbf{X}_{\mathcal{C}_i} \neq \emptyset$.

Durant la boucle while (ligne 3) l'assertion *"NbSol est le nombre d'extensions cohérentes de* $\mathcal{A}$ *sur* $X_{\mathcal{C}_i}$ *et la descendance de* $\mathcal{C}_i$ *avec les valeurs* v *de* X *déjà examinée"* est vérifiée. Nous allons montrer qu'elle est vraie à l'issue de la boucle. Soit v une valeur de X à examiner :

- $\mathcal{A} \cup \{X \leftarrow v\}$ est incohérente. Il n'y a pas d'extension possible, *NbSol* est inchangé, l'assertion est donc vérifiée.
- $\mathcal{A} \cup \{X \leftarrow v\}$ est cohérente. et $\mathcal{V}ar(\mathcal{C}_i, X_{\mathcal{C}_i} \setminus \{X\}) \subset \mathcal{V}ar(\mathcal{C}_i, X_{\mathcal{C}_i})$. L'hypothèse d'induction nous permet de déduire que #BTD$(\mathcal{A} \cup \{X \leftarrow v\}, \mathcal{C}_i, X_{\mathcal{C}_i} \setminus \{X\})$ renvoie le nombre d'extensions de $\mathcal{A}$ telles que $X = v$. Le résultat obtenu est ajouté à *NbSol*, l'assertion est vérifiée.

Après la boucle (ligne 5), #BTD$(\mathcal{A}, \mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i})$ renvoie le nombre d'extensions cohérentes de $\mathcal{A}$ sur $\mathbf{X}_{\mathcal{C}_i}$ et sur la descendance de $\mathcal{C}_i$. Donc $P(\mathcal{A}, \mathcal{V}ar(\mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i}))$ est vérifiée.

2^{nd} **cas :** $\mathbf{X}_{\mathcal{C}_i} = \emptyset$.

Durant la boucle while (ligne 1) l'assertion *"Pour chaque enfant de* $\mathcal{C}_i$ *déjà examiné il existe au moins une extension cohérente de* $\mathcal{A}$ *sur* $Desc(\mathcal{C}_i)$*. NbSol est le nombre d'extensions cohérentes de* $\mathcal{A}$ *sur les fils de* $\mathcal{C}_i$ *déjà examinés."*

Nous allons montrer qu'elle est vraie à l'issue de la boucle. Soit $\mathcal{C}_j$ un fils de $\mathcal{C}_i$.

- S'il existe un #good $g = (\mathcal{B}, nb)$ de $\mathcal{C}_i/\mathcal{C}_j$ tel que $\mathcal{A}[\mathcal{C}_i \cap \mathcal{C}_j] = \mathcal{B}$, par la définition 5.9 d'un #good $\mathcal{A}$ possède nb extensions sur $Desc(\mathcal{C}_j)$.
- S'il n'existe pas de #good $g = (\mathcal{B}, nb)$ de $\mathcal{C}_i/\mathcal{C}_j$ tel que $\mathcal{A}[\mathcal{C}_i \cap \mathcal{C}_j] = \mathcal{B}$ alors #BTD est appelée avec l'affectation $\mathcal{A}$ et $\mathcal{V}ar(\mathcal{C}_j, \mathcal{C}_j \setminus (\mathcal{C}_i \cap \mathcal{C}_j)) \subset \mathcal{V}ar(\mathcal{C}_i, \emptyset)$. L'hypothèse d'induction nous permet de déduire que #BTD$(\mathcal{A}, \mathcal{C}_j, \mathcal{V}ar(\mathcal{C}_j, \mathcal{C}_j \setminus (\mathcal{C}_i \cap \mathcal{C}_j)))$ renvoie nb le nombre d'extensions cohérentes que $\mathcal{A}$ possède sur $Desc(\mathcal{C}_i)$

Enfin la propriété 5.2 permet de conclure que $NbSol \leftarrow NbSol \times nb$ vérifie l'assertion. Si $NbSol = 0$ la boucle est terminée, car il n'existe pas d'extension cohérente de $\mathcal{A}$. Après la boucle (ligne 2) #BTD$(\mathcal{A}, \mathcal{C}_i, \emptyset)$ renvoie le nombre d'extensions cohérentes de $\mathcal{A}$ sur $Desc(\mathcal{C}_i)$. Donc $P(\mathcal{A}, \mathcal{V}ar(\mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i}))$ est vérifiée.

Pour résumer, l'algorithme #BTD satisfait la propriété $P(\mathcal{A}, \mathcal{V}ar(\mathcal{C}_i, \mathbf{X}_{\mathcal{C}_i}))$, et en particulier la propriété $P(\emptyset, \mathcal{V}ar(\mathcal{C}_1, \mathbf{X}_{\mathcal{C}_1}))$ pour le premier appel. □

#BTD a la même complexité que BTD.

Théorème 5.12 *#BTD a une complexité en $O(nmd^{w+1})$ en temps et $O(nsd^s)$ en espace avec n le nombre de clusters, d la taille du plus grand domaine, s la taille du plus grand séparateur, m est le nombre de contraintes et w la largeur de la décomposition arborescente traitée par #BTD.*

Démonstration.

Complexité en temps Dans le pire des cas, #BTD explore tous les clusters (au plus n) et essaye toutes les valeurs de chaque variable à l'intérieur de chaque cluster. A chaque instant au plus m contraintes sont vérifiées (*cf.* ligne 4 de l'algorithme). Grâce au principe d'enregistrement des #goods, il est impossible d'explorer un même cluster avec une même affectation deux fois. Le nombre d'affectations d'un cluster est borné par d^{w+1} avec $w = \max_{\mathcal{C}_i \in \mathbf{C}} \mid \mathcal{C}_i \mid -1$ la largeur de la décomposition arborescente. Par conséquent, #BTD a une complexité en temps en $O(nmd^{w+1})$.

Complexité en espace #BTD ne mémorise que les #goods. Ces affectations sur les intersections $\mathcal{C}_i \cap \mathcal{C}_j$ avec $\mathcal{C}_j$ un enfant de $\mathcal{C}_i$. Donc, si s est la taille de la plus grande intersection, #BTD a une complexité en espace en $O(nsd^s)$ car le nombre de ces intersections est majoré par n, le nombre de #goods pour une intersection est majoré par d^s et la taille d'un #good est au plus s.

□

En pratique, pour les problèmes ayant une grande largeur d'arbre, BTD explose en temps et en mémoire, voir à la section 5.4 page 108. Dans ce cas, nous sommes intéressés par une méthode d'approximation.

5.3 Calcul approché du nombre de solutions avec `Approx`#BTD

Nous considérons ici des CSP qui ne sont pas nécessairement structurés. Nous définissons une collection de sous-problèmes d'un problème donné $\mathcal{P}$ en partitionnant l'ensemble de ses contraintes. Dans le cas d'un CSP binaire, cela revient à partionner l'ensemble des arêtes de son graphe de contraintes.

Propriété 5.13 *Soit un graphe $G = (\mathbf{X}, \mathbf{A})$ associé au problème $(\mathbf{X}, \mathbf{D}, \mathbf{C})$. Il existe un partitionnement $\{\mathbf{C}_1, \ldots, \mathbf{C}_k\}$ de $\mathbf{C}$ tel que*

1. *Pour chaque sous-problème $(\mathbf{X}_i, \mathbf{D}_i, \mathbf{C}_i)$ [5] admet un graphe de contraintes $G_i = (\mathbf{X}_i, \mathbf{A}_i)$ triangulé où $\mathbf{A}_i \subseteq \mathbf{A}$ est l'ensemble des arêtes associées à $\mathbf{C}_i$.*
2. $\bigcup_{i=1}^{k} \mathbf{X}_i = \mathbf{X}$, $\bigcup_{i=1}^{k} \mathbf{C}_i = \mathbf{C}$ *et* $\bigcap_{i=1}^{k} \mathbf{C}_i = \emptyset$.

5. Les variables $\mathbf{X}_i \subseteq \mathbf{X}$ sont celles présentes dans les contraintes de $\mathbf{C}_i$

Justifications. Soit la partition $\{\mathbf{C}_1, \ldots, \mathbf{C}_m\}$ telle que $\forall i\ \mathbf{C}_i = \{c_i \in \mathbf{C}\}$. Chaque graphe de contraintes est une clique formée des sommets associés aux variables de la contrainte. Chacun d'entre eux est donc triangulé. Par construction la deuxième condition est également vérifiée. □

Supposons que $\mathcal{S}_{\mathcal{P}_i}$ soit le nombre de solutions pour chaque sous-problème $\mathcal{P}_i$, $1 \leqslant i \leqslant k$. Nous estimerons le nombre de solutions de $\mathcal{P}$ en exploitant la propriété suivante. Notons $\mathbb{P}(\mathcal{A} \models \mathcal{P})$ la probabilité de l'événement "$\mathcal{A}$ *est une solution de* $\mathcal{P}$". $\mathbb{P}(\mathcal{A} \models \mathcal{P}) = \dfrac{\mathcal{S}_{\mathcal{P}}}{\prod_{X \in \mathbf{X}} D_X}$.

Propriété 5.14 *Soit un CSP donné* $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{C})$ *et une partition* $\{\mathcal{P}_1, ..., \mathcal{P}_k\}$ *de* $\mathcal{P}$ *induite par une partition de* $\mathbf{C}$ *en* k *éléments.*

$$\mathcal{S}_{\mathcal{P}} \approx \left\lceil \left(\prod_{i=1}^{k} \frac{\mathcal{S}_{\mathcal{P}_i}}{\prod_{X \in \mathbf{X}_i} D_X} \right) \times \prod_{X \in \mathbf{X}} D_X \right\rceil$$

Démonstration.

$$\begin{aligned} \mathbb{P}(\mathcal{A} \models \mathcal{P}) &= \mathbb{P}(\mathcal{A} \models \mathcal{P}_1, \mathcal{A} \models \mathcal{P}_2, \ldots, \mathcal{A} \models \mathcal{P}_k) \\ &= \mathbb{P}(\mathcal{A} \models \mathcal{P}_1) \cdot \mathbb{P}(\mathcal{A} \models \mathcal{P}_2 \mid \mathcal{A} \models \mathcal{P}_1) \ldots \mathbb{P}(\mathcal{A} \models \mathcal{P}_k \mid \mathcal{A} \models \mathcal{P}_1, \ldots, \mathcal{A} \models \mathcal{P}_{k-1}) \end{aligned}$$

Si nous supposons l'indépendance entre les problèmes $\mathcal{P}_i$ nous avons

$$\begin{aligned} \mathbb{P}(\mathcal{A} \models \mathcal{P}) &\approx \mathbb{P}(\mathcal{A} \models \mathcal{P}_1) \cdot \mathbb{P}(\mathcal{A} \models \mathcal{P}_2) \ldots \mathbb{P}(\mathcal{A} \models \mathcal{P}_k) \\ &\approx \prod_{i=1}^{k} \mathbb{P}(\mathcal{A} \models \mathcal{P}_i) \\ &\approx \prod_{i=1}^{k} \left(\frac{\mathcal{S}_{\mathcal{P}_i}}{\prod_{X \in \mathbf{X}_i} D_X} \right) \end{aligned}$$

or $\mathcal{S}_{\mathcal{P}} = \mathbb{P}(\mathcal{A} \models \mathcal{P}) \times \prod_{x \in X} d_x$ d'où

$$\mathcal{S}_{\mathcal{P}} \approx \left(\prod_{i=1}^{k} \frac{\mathcal{S}_{\mathcal{P}_i}}{\prod_{X \in \mathbf{X}_i} D_X} \right) \times \prod_{X \in \mathbf{X}} D_X$$

Puisqu'un nombre de solutions est par nature un nombre entier c'est pourquoi nous prenons l'entier supérieur de la valeur de l'expression obtenue. □

Remarque 5.15 *L'approximation renvoie une réponse exacte si tous les sous-problèmes sont indépendants* ($\cap \mathbf{X}_i = \emptyset$) *ou* $k = 1$ *(*$\mathcal{P}$ *est déjà triangulé) ou s'il existe un sous-problème incohérent* $\mathcal{P}_i$ *(car*

dans ce cas l'algorithme renvoie zéro). De plus, nous pouvons fournir un majorant trivial du nombre de solutions dû au fait que chaque sous-problème $\mathcal{P}_i$ est une relaxation de $\mathcal{P}$ (le même argument est utilisé dans [Pesant, 2005] pour construire un majorant).

$$\mathcal{S}_{\mathcal{P}} \leq \min_{i \in [1,k]} \left\lceil \frac{\mathcal{S}_{\mathcal{P}_i}}{\prod_{x \in X_i} d_x} \times \prod_{x \in X} d_x \right\rceil \tag{5.1}$$

`Approx`#BTD est décrit par l'algorithme 7, appliqué à un problème $\mathcal{P}$ avec le graphe de contraintes $(\mathbf{X}, \mathbf{A})$, la méthode construit une partition $\{\mathbf{C}_1, \ldots, \mathbf{C}_k\}$ de $\mathbf{C}$ telle que chaque graphe de contraintes $(\mathbf{X}_i, \mathbf{A}_i)$ soit triangulé pour tout $1 \leqslant i \leqslant k$. Chaque sous-graphe partiel est produit par l'algorithme `MaxChord`$^+$ [Dearing et al., 1988] décrit par l'algorithme 6 [6]. Le sous-problème $(\mathbf{X}_i, \mathbf{D}_i, \mathbf{C}_i)$ est résolu par #BTD. Cette méthode renvoie une approximation du nombre de solutions de $\mathcal{P}$ basée sur la propriété 5.14 page précédente.

`MaxChord`$^+$ construit un sous-graphe partiel maximal triangulé, il est décrit par l'algorithme 6. Il choisi (étudie) une variable dont un maximum d'arêtes est déjà ajouté au sous-graphe partiel (ligne 2). Pour chaque voisin (dans le graphe d'origine) non encore étudié, l'algorithme ajoute l'arête, entre ce voisin et la variable étudiée, au sous-graphe partiel seulement si elle forme une clique (ligne 1). Ce processus est itéré jusqu'à ce que toutes les variables soient étudiées. L'ordre dans lequel les variables ont été sélectionnées forme un schéma parfait d'élimination inverse pour le sous-graphe partiel.

Exemple 5.16 *Soit le* CSP $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{C})$ *dont le graphe de contraintes est donné par la figure 5.4a, avec les variables* $\{X_1, \ldots, X_6\}$ *et les contraintes* $\{C_{12}, C_{14}, C_{16}, C_{23}, C_{345}, C_{56}\}$*, la contrainte ternaire* C_{345} *est représentée en gras dans le graphe de contraintes. L'application de la première partie de l'algorithme* `MaxChord`$^+$ *permet d'obtenir le sous-graphe partiel maximal triangulé de la figure 5.4b, la seconde partie de l'algorithme ne peut pas sélectionner la contrainte* C_{345} *car toutes les arêtes ne sont pas présentes dans le graphe de la figure 5.4b, de plus l'arête* $\{X_3, X_5\}$ *ne correspond à aucune contrainte, donc finalement le sous-graphe partiel 5.4b, donné par la figure 5.4c, est triangulé et peut être associé à un sous-problème de* $\mathcal{P}$*.*

Théorème 5.17 *L'algorithme* `Approx`#BTD *est correct, complet et termine.*

Démonstration. Il suffit de prouver que nous avons une partition des contraintes à la fin de la boucle while pour pouvoir appliquer la propriété 5.14 page précédente. Il est facile de montrer par induction en utilisant les invariants $\mathbf{X} = \mathbf{X}' \cup \left(\bigcup_{j=1}^{i} \mathbf{X}_j \right)$ et "$Q = (\mathbf{C}', \mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_i)$ est une partition de $\mathbf{C}$" dans la boucle.

6. Il construit un sous-graphe partiel maximal, en terme d'arêtes, pour les CSP binaires, mais ne garantit pas de fournir le plus grand sous-graphe partiel triangulé. Pour les CSP non binaires, nous ne garantissons pas la maximalité du sous-graphe partiel car nous ajoutons seulement les contraintes incluses dans le sous-graphe (maximal) triangulé obtenu.

Algorithme 6: MaxChord$^+(\mathbf{X}, \mathbf{C})$

Entrée :
un ensemble de variables $\mathbf{X}$
un ensemble de contraintes $\mathbf{C}$
Pré-conditions :
$\mathbf{X}$ est induit par l'ensemble des variables des portées des contraintes $\mathbf{C}$
Post-relations :
renvoie le graphe de contraintes de $(\mathbf{X}', \mathbf{C}')$ qui est un sous-graphe partiel de celui de $(\mathbf{X}, \mathbf{C})$

$(\mathbf{X}, \mathbf{A}) \leftarrow$ le graphe de contraintes associé à $(\mathbf{X}, \mathbf{C})$;
`/* Algorithme MaxChord de [Dearing et al., 1988] */`
foreach $U \in \mathbf{X}$ **do** $\mathbb{Y}(U) \leftarrow \emptyset$;
Choisir $V_0 \in X$;
$\mathbf{S} \leftarrow \{V_0\}$;
$\mathbf{A}' \leftarrow \emptyset$;
$l \leftarrow \mid \mathbf{X} \mid$;
while $l > 1$ **do**
  forall the $U \in \mathbf{X} \setminus \mathbf{S}$ ***tel que*** $\{U, V_0\} \subseteq C \in \mathbf{C}$ **do**
1   **if** $\mathbb{Y}(U) \subseteq \mathbb{Y}(V_0)$ **then**
      $\mathbb{Y}(U) \leftarrow \mathbb{Y}(U) \cup \{V_0\}$;
      $\mathbf{A}' \leftarrow \mathbf{A}' \cup \{U, V_0\}$;
2   Choisir $V_0 = \text{argmax}_{V \in \mathbf{X} \setminus \mathbf{S}} \mid \mathbb{Y}(V) \mid$;
  $\mathbf{S} \leftarrow \mathbf{S} \cup \{V_0\}$;
  $l \leftarrow l - 1$;
`/* Partie supplémentaire : Sélection de toutes les contraintes de C incluses dans le sous-graphe partiel maximal` $(\mathbf{X}, \mathbf{A}')$ `*/`
$\mathbf{C}' \leftarrow \emptyset$;
foreach $C \in \mathbf{C}$ **do**
  if $\forall \{U, V\} \subseteq C,\ \{U, V\} \in \mathbf{A}'$ **then**
    $\mathbf{C}' \leftarrow \mathbf{C}' \cup \{C\}$;
    $\mathbf{X}' \leftarrow \mathbf{X}' \cup \{U, V\}$
return $(\mathbf{X}', \mathbf{C}')$;

Initialisation Considérons $i = 0$ Nous avons $\mathbf{X} = \mathbf{X}'$ qui vérifie bien $\mathbf{X} = \mathbf{X}' \cup \left(\bigcup_{j=1}^{0} \mathbf{X}_j \right) = \mathbf{X}' \cup \emptyset =$ $\mathbf{X}'$ et $\mathbf{C} = \mathbf{C}'$ donc $Q = (\mathbf{C}')$ est bien une partition de $\mathbf{C}$. La propriété est vérifiée.

Induction Supposons que pour tout $k < i$ la propriété d'induction est vérifiée. Après l'application de MaxChord$^+$ nous avons $\mathbf{X}_i \subset \mathbf{X}'$ et $\mathbf{C}_i \subset \mathbf{C}'$ Soit $\mathbf{X}''$ l'ensemble des variables sur lesquelles portent les contraintes de $\mathbf{C}'' = \mathbf{C}' \setminus \mathbf{C}_i$, ainsi $\mathbf{X}' = \mathbf{X}_i \cup \mathbf{X}''$ et $(\mathbf{C}'', \mathbf{C}_i)$ est une partition de $\mathbf{C}'$. Par hypothèse d'induction nous avons

Algorithme 7: Approx#BTD$(\mathbf{X}, \mathbf{C})$: integer

Entrée :
un ensemble de variables $\mathbf{X}$
un ensemble de contraintes $\mathbf{C}$
Pré-conditions :
$(\mathbf{X}, \mathbf{D}, \mathbf{C})$ un CSP sur des domaines finis
Post-relations :
renvoie une approximation du nombre de solutions du CSP $(\mathbf{X}, \mathbf{D}, \mathbf{C})$ vérifiant la propriété 5.14

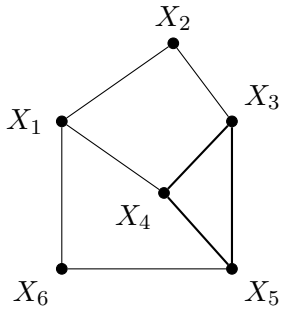
$(\mathbf{X}', \mathbf{C}') \leftarrow (\mathbf{X}, \mathbf{C})$;
$i \leftarrow 0$;
while $(\mathbf{X}', \mathbf{C}') \neq (\emptyset, \emptyset)$ **do**
  $i \leftarrow i + 1$;
  $(\mathbf{X}_i, \mathbf{C}_i) \leftarrow$`MaxChord`$^+(\mathbf{X}', \mathbf{C}')$;
  soit $\mathcal{P}_i$ *le sous-problème associé à* $(\mathbf{X}_i, \mathbf{C}_i)$;
  $\mathcal{S}_{\mathcal{P}_i} \leftarrow$`#BTD`$(\emptyset, \mathcal{C}_1^i, \mathcal{C}_1^i)$ avec $\mathcal{C}_1^i$ le cluster racine de la décomposition arborescente de $\mathcal{P}_i$;
  $(\mathbf{X}', \mathbf{C}') \leftarrow (\mathbf{X}'', \mathbf{C}' \setminus \mathbf{C}_i)$ avec $\mathbf{X}''$ l'ensemble des variables induites par $\mathbf{C}' \setminus \mathbf{C}_i$;
$k \leftarrow i$;
return $\left\lceil \left(\prod_{i=1}^{k} \frac{\mathcal{S}_{\mathcal{P}_i}}{\prod_{x \in X_i} d_x} \right) \times \prod_{x \in X} d_x \right\rceil$;

$$
\begin{aligned}
\mathbf{X} = \; & \mathbf{X}' \cup \bigcup_{j=1}^{i-1} \mathbf{X}_j \\
= \; & \mathbf{X}'' \cup \bigcup_{j=1}^{i} \mathbf{X}_j && \text{car } \mathbf{X}' = \mathbf{X}_i \cup \mathbf{X}'' \\
= \; & \mathbf{X}' \cup \left(\bigcup_{j=1}^{i} \mathbf{X}_j \right) && \text{car } \mathbf{X}' \leftarrow \mathbf{X}''
\end{aligned}
$$

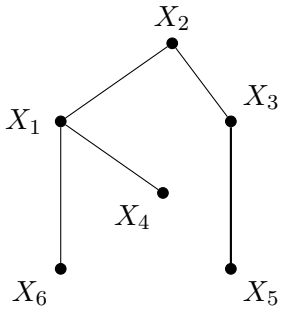
$$
\begin{aligned}
Q = \; & (\mathbf{C}', \mathbf{C}_1, \ldots, \mathbf{C}_{i-1}) \text{ partition de } \mathbf{C} \\
= \; & (\mathbf{C}'', \mathbf{C}_i, \mathbf{C}_1, \ldots, \mathbf{C}_{i-1}) \text{ partition de } \mathbf{C} && \text{car } (\mathbf{C}'', \mathbf{C}_i) \text{ est une partition de } \mathbf{C}'. \\
= \; & (\mathbf{C}', \mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_i) \text{ partition de } \mathbf{C} && \text{car } \mathbf{C}' \leftarrow \mathbf{C}''
\end{aligned}
$$

Finalement $(\mathbf{C}_1, \ldots, \mathbf{C}_k)$ est une partition de $\mathbf{C}$, nous pouvons appliquer le théorème 5.14. □

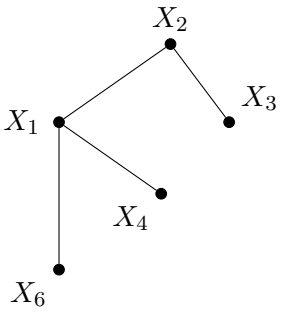
Théorème 5.18 Approx#BTD *a une complexité temporelle en* $\mathcal{O}(nm^2 d^{w'+1})$ *et spatiale en* $\mathcal{O}(ns'd^{s'})$ *avec* $s' < w' + 1 \leq c \leq w + 1 \leq n$ *où* $w' = \max_{\substack{\mathcal{C}_u^i \in \mathcal{C}^i \\ i \in [1,k]}} | \mathcal{C}_u^i | - 1$ *est la plus grande largeur d'arbre des sous-problèmes,* $s' = \max_{\substack{\mathcal{C}_u^i,\ \mathcal{C}_v^i \in \mathcal{C}^i \\ u \neq v \\ i \in [1,k]}} | \mathcal{C}_u^i \cap \mathcal{C}_v^i |$ *est la plus grande intersection de clusters et c est la taille de la clique maximale.*



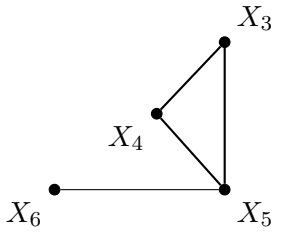
(a) Le graphe de contraintes de $\mathcal{P}$



(b) Le sous-graphe partiel triangulé maximal



(c) Le graphe de contraintes résultant pour le sous-problème $\mathcal{P}_1$



(d) Le graphe de contraintes résultant pour le sous-problème $\mathcal{P}_2$

Figure 5.4 – Graphes résultants de l'application de l'algorithme `MaxChord`$^+$, pour le CSP $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{C})$ avec $\mathbf{X} = \{X_1, \ldots, X_6\}$ et $\mathbf{C} = \{C_{12}, C_{14}, C_{16}, C_{23}, C_{345}, C_{56}\}$. $\mathcal{P}$ se partitionne en deux sous-problèmes structurés.

Démonstration.

Complexité en temps Le nombre d'itérations de Approx#BTD est inférieur à m. Chaque sous-graphe partiel triangulé et sa décomposition arborescente associée peuvent être calculés en $O(nm)$ [Dearing et al., 1988]. Alors, la complexité en temps de Approx#BTD est en $\mathcal{O}(nm^2d^{w'+1})$ avec la plus grande largeur d'arbre des sous-problèmes $w' = \max_{\substack{\mathcal{C}^i_u \in \mathcal{C}^i \\ i \in [1,k]}} | \mathcal{C}^i_u | -1$. Car chaque $\mathcal{P}_i$ est un sous-graphe partiel triangulé de $\mathcal{P}$, sa largeur d'arbre w' est égale à la clique maximale de son sous-graphe [Fulkerson and Gross, 1965] qui est par définition plus petite que la clique maximale du problème d'origine, appelé *clique number* c, inférieur à la largeur d'arbre du problème $w+1$.

Complexité en espace Approx#BTD à la même complexité que BTD appliqué sur le plus grand sous-problème $\mathcal{P}_i$ par rapport à la plus grande intersection de clusters $s' = \max_{\substack{\mathcal{C}^i_u,\, \mathcal{C}^i_v \in \mathcal{C}^i \\ u \neq v \\ i \in [1,k]}} | \mathcal{C}^i_u \cap \mathcal{C}^i_v |$.

□

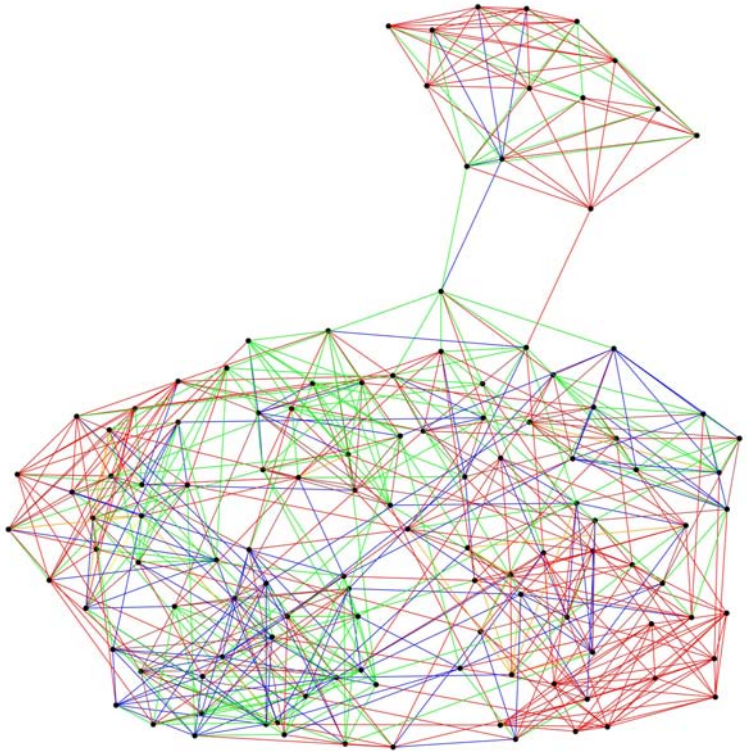


Figure 5.5 – Le graphe de contraintes de l'instance *games120*, ayant 6 parties formant des sous-graphes triangulés. La largeur d'arbre du graphe est w=41, après décomposition en 6 sous-problèmes la plus grande des largeurs d'arbre est w'=8.

5.4 Evaluation expérimentale

Les algorithmes #BTD et `Approx`#BTD ont été implémentés dans TOULBAR2 [7].

Pour #BTD, nous utilisons l'arc cohérence généralisée, lorsque les contraintes n'ont plus que deux ou trois variables non affectées, au lieu de backward checking pour des raisons d'efficacité. A l'intérieur des clusters, l'ordre dynamique, *min domain/max degree*, d'affectation des variables est modifiée par l'heuristique *conflict backjumping* [Lecoutre et al., 2006]. Nos méthodes exploitent un schéma de branchement binaire, chaque variable est affectée à sa première valeur ou celle-ci est effacée du domaine.

Nous avons testé #BTD et `Approx`#BTD sur des benchmarks SAT [8] et CSP [9]. Pour SAT, nous avons sélectionné des instances académiques (random k-SAT *wff*, All-Interval Series *ais*, tour de Hanoi *hanoi*) et industrielles (Circuit Fault Analysis *ssa* et *bit*, logistique *logistics*) satisfaisables. Pour CSP, nous avons sélectionné des instances de coloriage de graphes (comptage du nombre de solutions optimales).

7. version 0.8 pour ces expérimentations
8. www.satlib.org, www.satcompetition.org.
9. mat.gsia.cmu.edu/COLOR02/.

Pour les solveurs #SAT, les instances CSP sont traduites en SAT en utilisant la transformation directe (une variable booléenne par valeur des domaines, une clause par domaine pour qu'au moins une valeur soit prise et un ensemble de clauses binaires pour interdire la sélection de plusieurs valeurs).

Les expérimentations ont été réalisées sur un ordinateur 2.6GHz Intel Xeon avec 4GO sous Linux 2.6. Les temps CPU sont en secondes et limités à une heure. Pour les logiciels qui ne permettent pas de mesurer le temps CPU nous avons utilisé la commande bash `time`. Le temps mesuré par #BTD et `Approx`#BTD n'inclue pas le calcul de l'ordre d'élimination.

5.4.1 Evaluation des méthodes exactes

Nous avons comparé #BTD avec les solveurs `Relsat` [Bayardo and Pehoushek, 2000] v2.02, `Cachet`[Sang et al., 2004] v1.22 (avec une limite mémoire de 5 MO), `sharp`SAT [Thurley, 2006] v1.1 et `c2d` [Darwiche, 2004] v2.20 (avec une mémoire limite de 512 MO et une limite de 64 MO pour stocker les d-DNNF). Les méthodes #BTD et `c2d` utilisent l'heuristique *Min-Fill* d'ordre d'élimination de variables (excepté pour *hanoi* où nous avons utilisé l'ordre par défaut du fichier) pour construire la décomposition arborescence / les d-DNNF.

Nos résultats sont reportés dans la table 5.1. Nous remarquons que #BTD peut résoudre des instances avec une petite largeur d'arbre (excepté pour *ais* et *le450* qui ont peu de solutions). Les solveurs exacts de #SAT sont généralement meilleurs que #BTD pour les instances SAT (sauf pour *hanoi5*). Mais sur les instances CSP ces solveurs rencontrent des difficultés. Ici, #BTD avec le maintien de l'arc cohérence est plus performant que les solveurs #SAT qui utilisent la propagation unitaire.

5.4.2 Evaluation des méthodes approchées

La table 5.2 donne une analyse d'`Approx`#BTD sur les instances testées. Notre méthode de calcul approché exploite une partition du graphe de contraintes de telle sorte que les sous-problèmes résultants à résoudre ont une petite largeur d'arbre pour ces instances ($w' \leqslant 26$). En pratique, ça lui permet d'être relativement rapide connaissant la largeur d'arbre originelle. La borne supérieure du nombre de solutions donnée par `Approx`#BTD est généralement de mauvaise qualité surtout pour les instances SAT même lorsque les instances sont décomposées en peu de sous-problèmes (par exemple *ssa*).

Nous avons également comparé `Approx`#BTD avec la méthode de calcul approché `SampleCount`[Gomes et al., 2007]. Avec les paramètres ($s = 20$, $t = 7$, $\alpha = 1$), `SampleCount`-LB calcule une borne inférieure du nombre de solutions avec un intervalle de confiance. Avec les paramètres ($s = 20$, $t = 1$, $\alpha = 0$), `SampleCount`-A [10] nous donne seulement une approximation du nombre de solutions sans garantie, après la première itération de `SampleCount`-LB.

10. Avec ces paramètres `SampleCount`-A correspond à la méthode `ApproxCount` [Wei and Selman, 2005].

$\mathcal{P}$	n (Vars Bool)	d	w	$\mathcal{S}_\mathcal{P}$	c2d	sharpSAT	Cachet	Relsat	#BTD
					Time	Time	Time	Time	Time
SAT									
wff.3.100.150	100		39	1.8e21	*	mem	-	-	mem
wff.3.150.525	150		92	1.4e14	*	mem	**266.**	2509	mem
wff.4.100.500	100		80	-	*	mem	-	-	mem
ssa7552-038	1501		25	2.84e40	0.15	**0.06**	0.22	67	0.65
ssa7552-158	1363		9	2.56e31	0.10	**0.03**	0.07	3	0.19
ssa7552-159	1363		11	7.66e33	0.09	**0.04**	0.07	4	0.27
ssa7552-160	1391		12	7.47e32	0.12	**0.04**	0.08	5	0.30
2bitcomp_5	125		36	9.84e15	0.43	**0.05**	0.14	1	16.24
2bitmax_6	252	2	58	2.10e29	17.00	**0.87**	1.51	20	mem
ais6	61		41	24	0.05	**0.01**	0.03	< 1	0.08
ais8	113		77	40	0.51	**0.17**	0.58	< 1	3.27
ais10	181		116	296	16.64	**4.13**	29.19	6	543
ais12	265		181	1328	1147	**161.**	2173	229	-
logistics.a	828		116	3.8e14	-	**0.17**	3.78	10	mem
logistics.b	843		107	2.3e23	-	**1.38**	12.34	433	mem
hanoi4	718		46	1	7.18	**1.11**	32.64	3	1.87
hanoi5	1931		58	1	-	mem	-	-	**26.75**
CSP (Graph Coloring)									
2-Insertions_3	37 (148)	4	9	6.84e13	235.	mem	-	-	**7.80**
2-Insertions_4	149 (596)	4	38	-	-	mem	-	-	-
DSJC125.1	125 (625)	5	65	-	-	mem	-	-	mem
games120	120 (1080)	9	41	-	-	mem	-	-	mem
GEOM30a	30 (180)	6	6	4.98e14	0.86	5.53	-	-	**0.10**
GEOM40	40 (240)	6	5	4.1e23	1.00	mem	-	-	**0.09**
le450_5a	450 (2250)	5	315	3840	-	**32.31**	318	326	1100
le450_5b	450 (2250)	5	318	120	-	**13.12**	227	187	1364
le450_5c	450 (2250)	5	315	120	-	**2.18**	19.09	57	47.53
le450_5d	450 (2250)	5	299	960	-	**4.40**	14.60	36	92.03
mug100_1	100 (400)	4	3	1.3e37	0.19	23.88	-	-	**0.02**
myciel5	47 (282)	6	21	-	-	mem	-	-	mem

Table 5.1 – Comparaison des méthodes exactes. Légende : mem : dépassement en mémoire, - : dépassement en temps (pour c2d, * : dépassement en mémoire pour le stockage des NNF). Les colonnes sont : le nom de l'instance, le nombre de variables (ainsi que le nombre de variables booléennes pour les instances CSP transformées), la taille maximale des domaines, la largeur d'arbre de la décomposition arborescente, le nombre exact de solutions (si nous le connaissons), le temps pour c2d, sharpSAT, Cachet, Relsat, #BTD.

Les résultats de cette comparaison sont donnés par la table 5.3. La qualité de l'approximation trouvée par `Approx#`BTD est relativement bonne et elle est comparable (sauf pour les instances *ssa*, *logistics*, *mycielб-7*) à `SampleCount-A` qui prend le plus souvent (beaucoup) plus de temps.

5.5 Conclusion

Dans ce chapitre nous avons adapté une méthode structurelle de résolution des CSP au problème de comptage de solutions dans les CSP. Cependant la nature du problème de comptage rend ce problème difficile et les approches exactes montrent une certaine inefficacité à le résoudre, c'est pourquoi nous nous sommes intéressés à une décomposition structurelle des problèmes permettant de calculer le nombre exact de solutions des sous-problèmes induits par cette décomposition et nous permettre d'obtenir un calcul approché du nombre de solutions du problème global.

Ce travail pourra être étendu vers le calcul des solutions optimales d'un WCSP. Les #goods définis devront intéger la valeur du coût des solutions. Ce travail pourra s'inspirer de l'adaptation de BTD pour la résolution de CSP aux WCSP [de Givry et al., 2006].

Récemment dans [Rollon et al., 2011] les auteurs se sont intéressés au calcul des k meilleures solutions d'un WCSP. Ces k meilleures solutions pouvant être les solutions optimales et les suivantes (si le nombre de solutions optimales est inférieur à k). La question des k meilleures solutions a son intérêt dans l'étude générale de la reconstuction d'haplotypes. Par exemple dans le problème de reconstruction d'haplotypes, que nous aborderons dans la partie 3, les généticiens sont non seulement intéressés par la meilleure solution mais par de l'ensemble des meilleures solutions [11] ou des solutions optimales ou plus généralement par la distribution (de probabilité) de toutes les solutions.

Par ailleurs, dans les réseaux de Markov le calcul de la constante de normalisation, consistant à sommer sur toutes les valeurs de toutes les variables le produit des fonctions du réseau, s'apparente au problème de comptage. Après l'adaptation aux WCSP de nos méthodes de comptage (exacte et surtout approchée) l'étape suivante de ce travail sera d'étudier son utilisation pour ce calcul de constante de normalisation.

11. En effet, l'exactitude d'un phénomène étudié n'est pas forcément expliquée par une solution optimale du modèle utilisé.

$\mathcal{P}$	n	d	$\mathcal{S}_\mathcal{P}$	w	w'	k	$\hat{\mathcal{S}}_\mathcal{P}$			Time
SAT										
wff.3.100.150	100		1.80e21	39	3	6	$\approx$ 3.10e21	$\leq$	5.05e27	0.03
wff.3.150.525	150		1.14e14	92	3	13	$\approx$ 1.58e15	$\leq$	1.13e42	0.18
wff.4.100.500	100		-	80	6	48	$\approx$ 1.59e16	$\leq$	4.87e29	0.47
ssa7552-038	1501		2.84e40	25	6	4	$\approx$ 9.33e38	$\leq$	3.79e142	1.34
ssa7552-158	1363		2.56e31	9	5	3	$\approx$ 2.22e25	$\leq$	1.41e79	0.77
ssa7552-159	1363		7.66e33	11	5	3	$\approx$ 6.53e27	$\leq$	6.33e88	0.85
ssa7552-160	1391		7.47e32	12	5	3	$\approx$ 4.50e26	$\leq$	3.36e106	1.09
2bitcomp_5	125		9.84e15	36	6	4	$\approx$ 8.61e16	$\leq$	6.81e27	0.02
2bitmax_6	252	2	2.10e29	58	6	4	$\approx$ 4.53e29	$\leq$	7.19e45	0.10
ais6	61		24	41	12	8	$\approx$ 1	$\leq$	1.81e9	0.04
ais8	113		40	77	16	11	$\approx$ 1	$\leq$	3.49e15	0.20
ais10	181		296	116	21	23	$\approx$ 1	$\leq$	2.06e22	0.84
ais12	265		1328	181	26	23	$\approx$ 1	$\leq$	3.19e29	2.47
logistics.a	828		3.8e14	116	10	24	$\approx$ 1	$\leq$	4.91e180	14.85
logistics.b	843		2.3e23	107	13	25	$\approx$ 1	$\leq$	2.15e169	14.08
hanoi4	718		1	46	10	8	$\approx$ 1	$\leq$	4.26e106	1.40
hanoi5	1931		1	58	12	11	$\approx$ 1	$\leq$	4.48e309	16.05
CSP (Graph Coloring)										
2-Insertions_3	37	4	6.84e13	9	1	3	$\approx$ 1.91e13	$\leq$	6.00e17	0.01
2-Insertions_4	149	4	-	38	1	6	$\approx$ 1.30e22	$\leq$	1.64e71	0.07
DSJC125.1	125	5	-	65	3	7	$\approx$ 1.23e13	$\leq$	2.27e70	0.12
games120	120	9	-	41	8	6	$\approx$ 1.12e78	$\leq$	1.92e99	9.84
GEOM30a	30	6	4.98e14	6	5	2	$\approx$ 7.29e14	$\leq$	1.81e15	0.04
GEOM40	40	6	4.1e23	5	5	2	$\approx$ 4.8e23	$\leq$	1.72e24	0.01
le450_5a	450	5	3840	315	4	13	$\approx$ 1	$\leq$	2.41e216	3.17
le450_5b	450	5	120	318	4	13	$\approx$ 1	$\leq$	5.72e213	3.23
le450_5c	450	5	120	315	4	20	$\approx$ 1	$\leq$	1.49e201	7.42
le450_5d	450	5	960	299	4	20	$\approx$ 1	$\leq$	8.58e200	7.38
mug100_1	100	4	1.3e37	3	2	2	$\approx$ 5.33e37	$\leq$	7.18e41	0.01
myciel5	47	6	-	21	1	8	$\approx$ 7.70e17	$\leq$	8.53e32	0.03
myciel6	95	7	-	35	1	13	$\approx$ 5.49e29	$\leq$	9.80e73	0.19
myciel7	191	8	-	66	1	21	$\approx$ 4.25e35	$\leq$	2.96e161	1.23

Table 5.2 – Analyse de la performance de `Approx`#BTD et des caractéristiques des sous-problèmes. Les colonnes sont : le nom des instances, le nombre de variables, la taille maximale des domaines, le nombre de solutions (s'il est connu), la largeur d'arbre de la décomposition arborescente du problème d'origine, la largeur d'arbre maximale des sous-problèmes, nombre de sous-problèmes dans la partition, nombre approché de solutions et la borne supérieure du nombre de solutions donnée par l'équation (5.1) et le temps de `Approx`#BTD.

$\mathcal{P}$	$\mathcal{S}_\mathcal{P}$	Approx#BTD		SampleCount-A		SampleCount-LB	
		$\hat{\mathcal{S}}_\mathcal{P}$	Time	$\hat{\mathcal{S}}_\mathcal{P}$	Time	$\hat{\mathcal{S}}_\mathcal{P}$	Time
SAT							
wff.3.100.150	1.8e21	≈ 3.10e21	0.1	≈ **1.37e21**	959.8	-	-
wff.3.150.525	1.4e14	≈ 1.58e15	0.2	≈ **3.80e14**	0.7	≥ 2.53e12	4.6
wff.4.100.500	-	≈ 1.59e16	0.5	≈ 4.15e16	2045.0	-	-
ssa7552-038	2.84e40	≈ 9.33e38	1.3	≈ **1.11e40**	134.0	≥ 3.54e38	1162.0
ssa7552-158	2.56e31	≈ 2.22e25	0.8	≈ **1.43e30**	14.1	≥ **1.43e30**	177.0
ssa7552-159	7.66e33	≈ 6.53e27	0.8	≈ 6.49e34	32.8	≥ **1.64e32**	182.0
ssa7552-160	7.47e32	≈ 4.50e26	1.1	≈ **5.08e32**	144.0	≥ 2.31e31	1293.0
2bitcomp_5	9.84e15	≈ 8.61e16	0.1	≈ **4.37e15**	0.2	≥ 3.26e15	1.2
2bitmax_6	2.10e29	≈ 4.53e29	0.1	≈ **1.62e29**	1.7	≥ 2.36e26	10.3
ais6	24	≈ 1	0.1	≈ **12**	0.1	≥ **12**	0.2
ais8	40	≈ 1	0.2	≈ **16**	1.5	≥ 12	15.9
ais10	296	≈ 1	0.8	≈ **124**	45.9	≥ 20	312.0
ais12	1328	≈ **1**	2.5	≈ 0	9.2	≥ 0	9.2
logistics.a	3.8e14	≈ 1	14.8	≈ **7.25e11**	171.0	≥ 0	605.0
logistics.b	2.3e23	≈ 1	14.1	≈ **2.13e23**	199.0	≥ 0	229.0
hanoi4	1	≈ **1**	1.4	≈ 0	5.2	≥ 0	5.2
hanoi5	1	≈ **1**	16.0	≈ 0	6.1	≥ 0	6.2
CSP (Graph Coloring)							
2-Insertions_3	6.84e13	≈ **1.91e13**	0.1	≈ 4.73e12	1.0	≥ 4.73e12	7.4
2-Insertions_4	-	≈ 1.30e22	0.1	≈ 0	3.8	≥ 0	3.8
DSJC125.1	-	≈ 1.23e13	0.1	≈ 0	73.1	≥ 0	73.2
games120	-	≈ 1.12e78	9.8	≈ 7.33e64	13.8	≥ 1.35e61	91.1
GEOM30a	4.98e14	≈ **7.29e14**	0.1	≈ 1.23e13	0.4	≥ 3.28e12	3.7
GEOM40	4.1e23	≈ **4.8e23**	0.1	≈ 2.14e20	1.5	≥ 6.50e19	9.3
le450_5a	3840	≈ **1**	3.2	≈ 0	8.6	≥ 0	8.6
le450_5b	120	≈ **1**	3.2	≈ 0	8.6	≥ 0	8.6
le450_5c	120	≈ **1**	7.4	≈ 0	110.0	≥ 0	111.0
le450_5d	960	≈ **1**	7.4	≈ 0	54.6	≥ 0	54.6
mug100_1	1.3e37	≈ **5.33e37**	0.1	≈ 4.2e34	2.1	≥ 4.20e34	15.6
myciel5	-	≈ 7.69e17	0.1	≈ 7.29e17	0.9	≥ 7.29e17	6.4
myciel6	-	≈ 5.49e29	0.2	≈ 9.38e40	4.5	≥ 7.42e36	30.7
myciel7	-	≈ 4.26e35	1.2	≈ 1.37e80	27.7	≥ 5.56e74	163.0

Table 5.3 – Comparaison des méthodes approchées. Légende : - : dépassement en temps (1 heure).

Partie III

Reconstruction d'haplotypes dans les pedigrees animaux

Notions élémentaires de génétique 6

6.1 Le génome

Tout être vivant contient dans ses cellules des éléments microscopiques porteurs de l'information génétique et de sa transmission au cours des générations : les *chromosomes*. Par exemple on trouve 46 chromosomes chez l'Homme, 64 chez le cheval, 60 chez la vache et 38 chez le chat.

Ces chromosomes sont formés de longues chaînes de molécules d'Acide DésoxyriboNucléique ou ADN. La molécule d'ADN est une double hélice composée de 2 brins enroulés l'un autour de l'autre et chacun de ces brins est constitué d'un enchainement de bases de nucléotides qui se fait dans un sens déterminé. L'appariement des 2 brins qui composent l'ADN est réalisé par des paires de bases : Adénine (A) - Thymine (T) et Guanine (G) - Cytosine (C). Il faut savoir que l'Adénine peut se lier à la Thymine (AT) et la Guanine à la Cytosine (GC) mais en aucun cas AG,AC,TG et TC. Grâce à l'alternance des 4 bases A,C,T,G toutes ces séquences constituent un message codé portant les informations génétiques.

Chaque chromosome est constitué de plusieurs dizaines, voire centaines de millions de paires de bases, chez l'Homme on en compte environ 4 milliards. Un *gène* est une partie d'un chromosome formant une unité d'information génétique. Il détermine la mise en place et la transmission de caractères observables (par exemple la couleur des yeux, le groupe sanguin...). Chacune des différentes formes ou versions d'un même gène, relative au même caractère est appelée *allèle*. Chaque individu possède un ensemble d'allèles qui lui est propre qui constitue son *génotype*. Le *locus* du gène est un emplacement physique précis et invariable sur le chromosome.

L'ensemble du matériel génétique d'une espèce, ensemble de gènes portés par les chromosomes, constitue le *génome*.

Il existe également des marqueurs génétiques, même s'il ne font pas partie de vrais gènes, car ils n'ont pas forcément de fonctionnalité, on parle également de génotypes et allèles. La transmission des marqueurs obéit aux mêmes règles que celles des gènes décrits dans la section suivante.

Les événements de mutation dans l'ADN sont les principaux facteurs responsables de la différence entre individus. Les SNP (*Single Nucleotide Polymorphism*) sont les mutations les plus communes. Ils sont le résultat de la mutation du nucléotide A en T (et inversement) ou de C en G (et inversement). Un SNP est défini comme une position sur le chromosome (son locus) lorsqu'un des deux nucléotides est observé dans au moins 10% de la population. Par exemple, nous observons un SNP lorsque la séquence AG**T**TCG est modifiée en AG**A**TCG. Les deux nucléotides impliqués dans un SNP correspondent aux allèles de ce SNP (dit bi-allélique).

Chez les espèces diploïdes telles que la plupart des espèces animales et végétales, les chromosomes sont associés par paire : on parle de *chromosomes homologues*, l'un provenant de la mère et l'autre du père. Ces chromosomes partagent le même ensemble de gènes mais avec des allèles pouvant être différents. Pour un locus donné un individu est *homozygote* si les deux allèles sont identiques sinon il est *hétérozygote*. Seuls les chromosomes sexuels dérogent à la règle, les chromosomes X et Y ne partagent pas exactement les mêmes gènes (on retrouve deux chromosomes X chez les femelles et les chromosomes X et Y chez les mâles). L'ensemble des allèles présents sur un même chromosome d'un individu est appelé un *haplotype*.

L'expression d'un caractère (par exemple physique) dû à un premier allèle peut masquer totalement celle dû au second allèle. Dans ce cas le premier est dit *dominant* et le second *récessif*. La *codominance* correspond à l'expression conjointe des deux allèles.

6.2 Les lois de Mendel

Gregor Mendel est reconnu comme étant le père fondateur de la génétique, il est à l'origine des *lois dites de Mendel* qui définissent comment les gènes se transmettent de génération en génération [Mendel, 1866].

Définition 6.1 Loi d'uniformité des individus de première génération
Lorsque l'on croise des individus de deux races pures (homozygotes pour tous les gènes), différant l'une de l'autre par un ou plusieurs allèles, les individus obtenus en première génération (F1) sont tous semblables entre eux : la première génération est homogène. (*cf.* figure 6.1 page suivante)

Définition 6.2 Loi de la ségrégation des allèles
La génération F2, celle obtenue en croisant entre eux les individus de la première génération entre eux, est hétérogène. Cette hétérogénéité traduit une ségrégation des allèles. Il en résulte que les gamètes (cellules sexuelles) ne portent qu'un seul allèle de chaque gène. (*cf.* figure 6.2 page ci-contre)

Par cette loi, un individu est déterminé par ses deux gènes, dont un est une copie d'un des allèles de la paire que possède sa mère et l'autre est une copie d'un des allèles de la paire que possède son père.

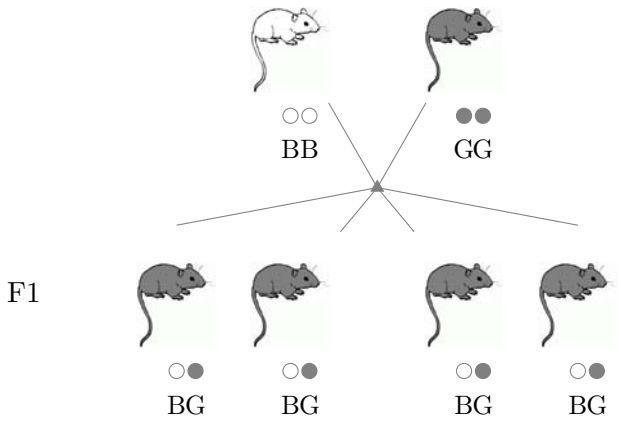


Figure 6.1 – Exemple de croisement entre une souris grise (GG caractère dominant) et une souris blanche (BB, caractère récessif). A la première génération (F1) toutes les souris sont grises (GB)

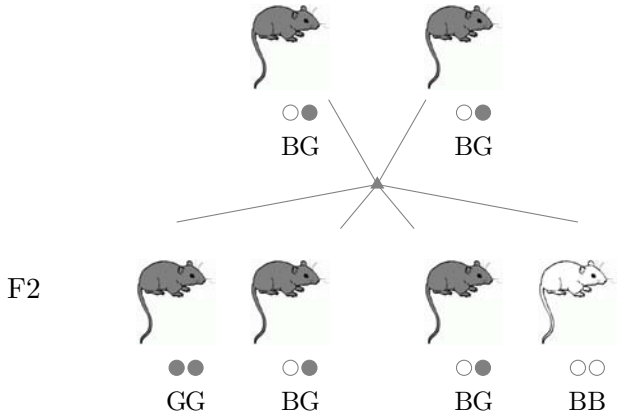


Figure 6.2 – Exemple de croisement de deux souris issues de la génération F1 de la figure précédente. A la deuxième génération (F2) on retrouve 75 % de souris grises (25% d'homozygotes GG et 50% d'hétérozygotes GB) et 25% de souris blanches (homozygotes BB)

Définition 6.3 Loi de l'indépendance des couples de gènes
Si l'on croise deux individus de race pures qui diffèrent par plusieurs gènes, on constate que ces caractères sont, dans les générations suivantes, hérités de façon indépendante les uns des autres, et se retrouvent associés en F2 comme s'ils avaient été distribués au hasard. Les différents gènes sont donc indépendants les uns des autres (gènes non liés). La disjonction se fait d'une manière indépendante pour les différents gènes.

Cette dernière loi n'est valable que si les gènes sont indépendants, par exemple sur des chromosomes différents. En effet si deux gènes sont proches sur un même chromosome, leur ségrégation n'est pas

indépendante. La section suivante détaille ce point.

6.3 Crossing-over et distance génétique

6.3.1 Crossing-over

Chaque cellule, des êtres vivants diploïdes, a une copie des n paires de chromosomes homologues. Seules les cellules sexuelles appelées *gamètes* ne possèdent que n chromosomes. Ainsi lors de la reproduction, les deux gamètes (celui de la mère et celui du père) fusionnent pour donner une nouvelle cellule possédant à nouveau $2n$ chromosomes. Cette cellule est à l'origine de toutes celles composant le nouvel individu créé. La production des gamètes se fait lors d'un processus de division cellulaire appelé *méiose*. Il crée quatre cellules à partir d'une cellule contenant $2n$ chromosomes. Cependant ces quatre cellules ne contiennent pas forcément un des deux chromosomes de chaque paire mais un mélange des deux. Lors de la méiose les chromosomes homologues se rapprochent, il arrive alors qu'ils s'échangent des parties. Ce phénomène est appelé *crossing-over*.

(a) Deux chromosomes homologues **(b)** Résultat de deux crossing-over en r_1 et r_2

Figure 6.3 – Échange entre deux chromosomes homologues lors de la méiose

La probabilité r qu'il y ait un crossing-over [1] entre deux loci donnés est appelé *taux de recombinaison*. Deux gènes sont dits indépendants si $r = \frac{1}{2}$ et liés si $r < \frac{1}{2}$ (par nature r ne peut être supérieur à $\frac{1}{2}$).

6.3.2 Distance génétique

On définit la *distance génétique* comme une fonction du taux de recombinaison. Sous l'hypothèse d'absence d'interférence entre les crossing-over cette distance s'appelle *distance de Haldane* [Haldane, 1919].

En supposant que les crossing-over sont indépendants (non interférence), tirés au hasard dans le chromosome et que leur nombre suit une loi de Poisson, la relation entre d_{kl} la distance génétique entre deux loci k et l et leur taux de recombinaison r_{kl} est donnée par

$$d_{kl} = -\frac{1}{2}\log(1 - 2r_{kl}) \qquad \text{définie pour } 0 \leqslant r_{kl} \leqslant \frac{1}{2} \tag{6.1}$$

1. ou plus précisément un nombre impair de crossing-over

et inversement

$$r_{kl} = \frac{1}{2}(1 - \exp(-2d_{kl})) \tag{6.2}$$

Pour de petites distances on peut remarquer que $d_{kl} \approx r_{kl}$. Ainsi nous pourrons supposer qu'un centiMorgan (notée 1cM) correspond à 1% de recombinaison entre les loci.

Il existe cependant d'autres distances génétiques comme la distance de Kosambi [Kosambi, 1944] par exemple. Mais dans la suite du mémoire j'ai utilisé exclusivement la distance de Haldane.

La *carte génétique* d'une espèce est une représentation du génome de l'espèce. Elle contient les loci (marqueurs) que l'on a pu identifier dans chaque chromosome, leur ordre relatif ainsi que la distance génétique qui les sépare. Il existe actuellement plusieurs logiciels dont Mapmaker [Lander et al., 1987], JoinMap [Stam, 1993] ou CartaGène [de Givry et al., 2005a] permettant la construction de cartes.

Les techniques actuelles permettent de génotyper plusieurs dizaines de milliers de SNP avec les puces [2] à ADN de 50 000 SNP utilisées pour le génotypage des bovins, par exemple.

6.4 Déséquilibre de liaison

La notion de déséquilibre de liaison entre deux allèles situés à des loci différents fait référence à la corrélation entre ces allèles. Le déséquilibre de liaison décrit une non-indépendance de ces allèles.

Au contraire, l'équilibre de liaison entre deux allèles décrit une indépendance entre eux. Ainsi nous avons pour l'allèle a_1 du locus l_1 et l'allèle a_2 du locus l_2 :

$$\texttt{freq}(a_1, a_2) = \texttt{freq}(a_1) \cdot \texttt{freq}(a_2) \tag{6.3}$$

où $\texttt{freq}(a_1, a_2)$, $\texttt{freq}(a_1)$, $\texttt{freq}(a_2)$ désignent respectivement les fréquences de l'haplotype (a_1, a_2), de l'allèle a_1 au premier locus et de l'allèle a_2 au second locus.

La force du déséquilibre de liaison entre deux locus est mesurée à l'aide du cœfficient de déséquilibre de liaison D suivant :

$$D(a_1, a_2) = \texttt{freq}(a_1, a_2) - \texttt{freq}(a_1) \cdot \texttt{freq}(a_2) \tag{6.4}$$

Une valeur de D positive traduit le fait que les deux allèles ont tendance à être présents ensemble ; une valeur négative traduit le fait inverse.

Propriété 6.4 *[Cierco-Ayrolles et al., 2004] Soit deux loci l_1 et l_2 bi-alléliques avec* a *et* b *leurs deux allèles possibles.*
$D(\mathrm{a}, \mathrm{a}) = -D(\mathrm{a}, \mathrm{b}) = -D(\mathrm{b}, \mathrm{a}) = D(\mathrm{b}, \mathrm{b})$

Il existe différentes mesures pour le déséquilibre de liaison, pour plus d'information à ce sujet la revue [Cierco-Ayrolles et al., 2004] les détaille.

2. puce éléctronique dédiée au génotypage

Reconstruction d'haplotypes : l'existant 7

La gestion des populations d'élevage passe par l'amélioration et la conservation des ressources génétiques. Cette tâche est aujourd'hui effectuée dans la plupart des cas sur des données du contrôle de performances (généalogie, performances). Par contre, les développements technologiques en génétique moléculaire et l'existence des marqueurs génétiques ont ouvert la possibilité d'utiliser une toute nouvelle source d'information.

Les marqueurs génétiques permettent d'explorer l'architecture génétique :

- pour trouver la localisation des zones du génome ayant un effet quantitatif sur des caractères d'intérêt (localisation de QTL, *Quantitive Trait Loci*)
- pour tracer les événements historiques de la population comme la sélection
- pour vérifier et contrôler la variabilité génétique.

Pour cela nous avons besoin de données sur la structure fine des chromosomes, notamment sur les marqueurs génétiques qui se situent ensemble sur le même chromosome dans la population : collection d'haplotypes de la population, mais il est également important de connaître, pour chaque individu, les différents marqueurs originels.

Avec les méthodes expérimentales actuelles, il nous est trop coûteux de mesurer les haplotypes des individus [Niu, 2004]. Seuls les génotypes sont mesurés avec un coût raisonnable. Cependant les haplotypes sont plus informatifs que les génotypes et dans certains cas permettent de mieux prédire la gravité d'une maladie ou d'expliquer un phénotype particulier. Il est donc important à partir de ces mesures génotypiques, de retrouver les haplotypes pouvant les expliquer au mieux : c'est le principe de la *reconstruction d'haplotypes*.

7.1 Reconstruction d'haplotypes : le principe

7.1.1 Un exemple

Dans cet exemple nous allons nous intéresser à une famille de cinq individus : un père, une mère et leurs trois enfants.

Nous considérons une paire de chromosomes homologues dont cinq marqueurs génétiques peuvent être mesurés, et sont identifiés par leur locus. Avec les techniques de mesures de génotypes nous avons obtenus les observations récapitulées dans le tableau suivant :

	Père	Mère	Fils A	Fils B	Fils C
locus 1	ab	ab	bb	ab	bb
locus 2	bb	ab	ab	bb	ab
locus 3	aa	aa	aa	aa	aa
locus 4	ab	ab	aa	bb	bb
locus 5	ab	bb	ab	bb	bb

Pour chaque individu, à chaque locus nous avons la valeur des deux allèles présents. Par exemple, pour le père nous avons les allèles a et b au locus 1 et deux fois l'allèle a au locus 3.

Le but de la reconstruction d'haplotypes va être d'identifier pour chacun des deux chromosomes homologues l'ensemble des cinq allèles présents. Il faut évidemment que les règles génétiques soient respectées. Ces règles sont :

1. A chaque locus, chaque allèle observé est présent sur un des chromosomes homologues.
2. A chaque locus, un individu reçoit un allèle de son père et un allèle de sa mère.
3. Le chromosome d'un individu est l'ensemble des allèles transmis par son père et son homologue est l'ensemble des allèles transmis par sa mère.

La figure 7.1 représente les chromosomes homologues de chaque individu visualisés verticalement et séparés par un trait. Elle illustre une reconstruction possible à partir des observations de génotypes que nous avons, vérifiant les trois points précédents. Nous avons identifié les chromosomes des parents par des couleurs (bleu et violet pour ceux du père, orange et rouge pour ceux de la mère). Pour le père le chromosome identifié par la couleur bleue porte l'allèle b au locus 1, pour respecter l'observation ab du tableau nous avons l'allèle a de ce même locus présent sur le chromosome homologue identifié par la couleur violette (la règle 1 est vérifiée). Dans cet exemple, nous n'avons pas d'information sur les parents du père et de la mère, les règles 2 et 3 Le père et la mère dans notre exemple n'ont pas de parents (connus) les règles 2 et 3 ne peuvent pas être prises en compte, pour eux ils nous suffit de vérifier la règle 1 [1].

1. Nous verrons plus tard qu'il existe d'autres informations pour guider la reconstruction de leurs haplotypes plus "intelligemment", toujours en respectant cette première règle.

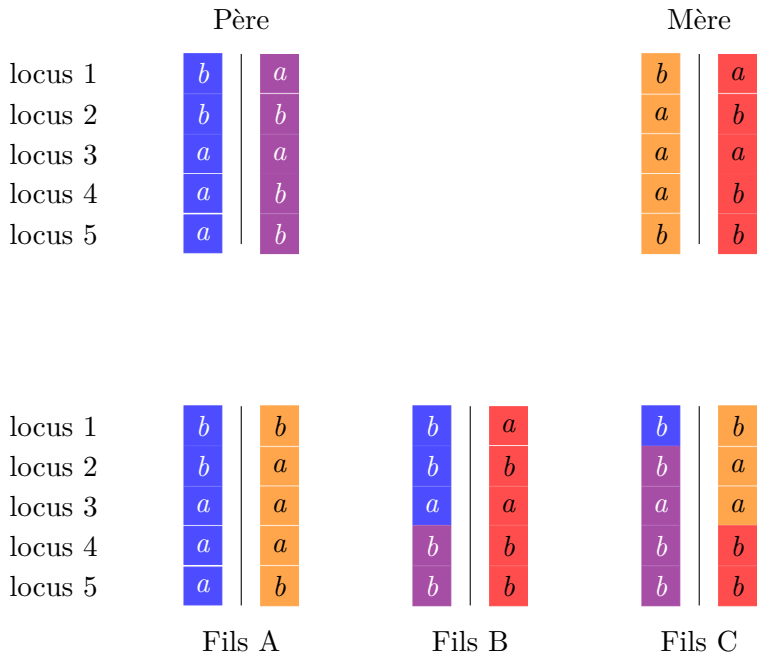


Figure 7.1 – Une reconstruction possible compatible avec les observations et les règles génétiques

Pour les trois fils, la règle 1 est bien respectée à la figure 7.1, intéressons nous maintenant à la vérification des règles 2 et 3 :

- Le respect de la règle 2 passe visuellement par le fait que pour chaque locus, un allèle doit être associé à la couleur bleue ou violette (signifiant que l'allèle a été transmis par le père) et l'autre à la couleur rouge ou orange (signifiant que l'allèle a été transmis par la mère).
- Le respect de la règle 3 correspond au fait qu'un des chromosomes doit être une sucession d'allèles bleus et violets (ensemble des allèles transmis par le père) et l'autre une succession d'allèles rouges et oranges (ensemble des allèles transmis par la mère).

Avec la reconstruction proposée nous pouvons observer d'une part que le fils A a reçu une copie complète du chromosome bleu de son père et une copie complète du chromosome orange de sa mère d'autres part que des événements de recombinaisons ont été identifiés (résultat de crossing over lors de la méiose) chez les fils B et C. Un changement de couleur dans les chromosomes symbolise ces événements [2]. Pour le fils B, nous pouvons en observer un entre le locus 3 et le locus 4 sur le chromosome (haplotype) transmis par le père. Il en va de même entre les loci 2 et 3 pour le chromosomse paternel du fils C et les loci 3 et 4 pour son chromosome maternel.

En résumé, pour cette reconstruction (figure7.1) nous avons identifié trois événements de recombinaisons (un chez le fils B et deux chez le fils C).

Cependant cette reconstruction n'est pas unique, en modifiant légèrement les haplotypes chez les parents (par exemple, inversion chez le père des allèles du locus 1) nous obtenons la figure 7.2 qui

2. Même principe utilisé pour présenter le crossing-over à la figure 6.3 page 120.

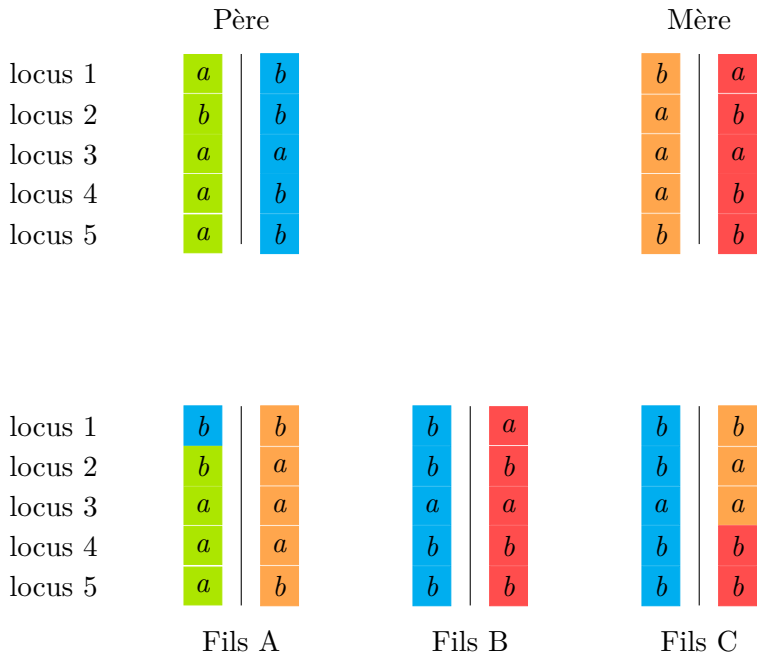


Figure 7.2 – Une autre reconstruction possible compatible avec les observations et les règles génétiques

propose une autre configuration respectant toujours nos trois règles. Avec cette reconstruction des haplotypes nous pouvons identifier deux événements de recombinaisons (une chez le fils A et une chez le fils C).

La section qui suit formalise tous ces concepts et nous verrons ensuite dans les sections suivantes les différents points de vue pour reconstruire les haplotypes des individus suivant des critères dont celui de minimiser le nombre de recombinaisons et ainsi préférer la solution de la figure 7.2 avec deux recombinaisons plutôt que la première avec ses trois événements de recombinaisons.

Les sections suivantes formalisent tous ces concepts et nous amèneront à considérer différents points de vue pour reconstruire les haplotypes des individus selon des critères comme celui de minimiser le nombre de recombinaisons et ainsi préférer la solution de la figure 7.2 avec deux recombinaisons seulement plutôt que celle de la figure 7.1 avec trois recombinaisons.

7.1.2 Une représentation formelle

Les différents allèles (deux dans le cas des SNP bi-alléliques) d'un locus sont représentés par des lettres, par la suite, nous utiliserons a et b.

A chaque locus l, un génotype G_l peut être observé, il prend sa valeur parmi aa, ab, bb. Il correspond à la paire non ordonnée des allèles, donc ab est identique à ba[3]. Pour un individu i, l'ensemble de ses

3. Dans ce cas nous avons choisi de les positionner suivant l'ordre lexicographique.

génotypes le long de L loci est noté $G^i = (G^i_1, \ldots, G^i_L)$.

L'individu i possède deux haplotypes H^P_i et H^M_i.

Chaque haplotype est une séquence de L allèles. A^P_{il} est l'allèle présent sur l'haplotype H^P_i au locus l, de même l'allèle A^M_{il} est sur l'haplotype H^M_i.

Le génotype et les haplotypes sont liés, à chaque locus l, $\{A^P_{il}, A^M_{il}\}$ forme le génotype du locus.

La paire (non ordonnée) de ses deux haplotypes forme son *diplotype* noté $\frac{H^P_i}{H^M_i}$. L'exemple suivant résume toutes les notations.

Exemple 7.1 *Soit un individu i admettant les génotypes suivants sur 5 loci :* aa ab aa bb ab

Pour exemple, pour le cinquième locus, nous avons $G^i_5 =$ ab*, les possibilités pour les haplotypes vont être* $(A^P_{i5} =$ a *et* $A^M_{i5} =$ b$)$ *ou* $(A^P_{i5} =$ b *et* $A^M_{i5} =$ a$)$.

Les deux haplotypes suivants expliquent le génotype de i : aaaba *et* ababb

Nous pouvons écrire $H^P_i =$ aaaba *et* $H^M_i =$ ababb

A ce stade, il est équivalent d'écrire $H^P_i =$ ababb *et* $H^M_i =$ aaaba*.*

Le diplotype de l'individu est alors $\frac{\text{aaaba}}{\text{ababb}}$*.*

La définition suivante formalise la notion de génotype "expliqué" par deux haplotypes.

Définition 7.2 Haplotypes compatibles
Soient un génotype $G = (G_1, \ldots, G_L)$ sur L loci et $H^P = (A^P_1, \ldots A^P_L)$, $H^M = (A^M_1, \ldots, A^M_L)$ deux haplotypes. H^P et H^M sont dit *compatibles* avec G si et seulement si

$$\forall\, 1 \leqslant l \leqslant L,\ G_l = \begin{cases} A^P_l A^M_l & \text{si } A^P_l \leqslant A^M_l \\ A^M_l A^P_l & \text{si } A^M_l \leqslant A^P_l \end{cases}$$

Définition 7.3 Reconstruction d'haplotypes
A partir d'un ensemble de n génotypes (sur plusieurs loci) $G = \{g_1, g_2, \ldots, g_n\}$ la reconstruction d'haplotypes permet d'obtenir pour chaque génotype g_i deux haplotypes h^1_i, h^2_i compatibles.

Exemple 7.4 *Soit le génotype* aa ab ab bb ab *sur 5 loci. Il y a quatre explications possibles pour ce génotype :* $\frac{\text{aaaba}}{\text{abbbb}}$, $\frac{\text{aabba}}{\text{ababb}}$, $\frac{\text{aabbb}}{\text{ababa}}$ *et* $\frac{\text{aaabb}}{\text{abbba}}$

Remarque 7.5 *Reconstruire un des deux haplotypes suffit à partir du génotype. Le second peut être immédiatement déduit. Ainsi dans l'exemple précédent, connaissant le génotype et l'haplotype* aaaba*, le seul haplotype possible pour former une paire compatible est* abbbb*. C'est pourquoi nous parlons de reconstruction d'haplotypes plutôt que de reconstruction de diplotypes.*

Cependant les critères de reconstruction qui vont être évoqués par la suite, sont à vérifier pour les deux haplotypes.

Il existe plusieurs approches pour le problème de reconstruction d'haplotypes. Certaines s'appliquent pour des individus n'ayant aucun lien de parenté dans une même population, ces approches sont présentées dans la section 7.2.1. D'autres approches présentées dans les sections 7.2.2 et 7.3 s'appliquent à des individus apparentés.

Dans le cas des individus apparentés nous avons l'information des liens de parenté sous forme de pedigree ou généalogie.

7.1.3 Pedigrees

Un pedigree ou une généalogie d'un groupe d'individus est défini par rapport à leurs relations de parenté. Ainsi un individu peut avoir des parents connus, c'est un individu *non-fondateur* du pedigree considéré ou au contraire être un individu *fondateur* et ainsi ne pas avoir de parents connus dans ce pedigree. De manière évidente, un individu a une mère et un père comme *parents*, et sera considéré comme leur *enfant* (fils ou fille). L'ensemble père-mère-enfants est une famille dite *nucléaire*. En effet la notion de famille dans les pedigrees est plus large, elle regroupe tous les individus ayant un lien de parenté plus ou moins éloigné dans le pedigree.

La définition proposée pour représenter un pedigree combine celles proposées dans [Thomas, 1985, Cannings et al., 1978, Lange and Elston, 1975]

Définition 7.6 Pedigree
Un pedigree est un graphe orienté bipartie $P = (N = I \cup M, A)$ où N est l'ensemble des sommets partitionné en deux ensembles I et M tels que I représente les individus (mâles et femelles) et M représente les *mariages* entre individus. L'ensemble des arcs A représente les mariages (arc d'un sommet individu vers sommet mariage) et les arcs descendances (un arc d'un sommet mariage vers sommet individu).

Dans ce mémoire, les sommets mâles sont représentés par des carrés et les sommets femelles par des ronds, les sommets mariages par des triangles. Lorsqu'on dessine les pedigrees il est classique de supprimer les directions des arcs car elles sont toujours orientées vers le bas, des parents vers les enfants via le sommet de mariage approprié. La figure 7.3 page ci-contre représente un pedigree avec les conventions présentées.

Les pedigrees dits complexes sont des généalogies présentant des boucles de consanguinité, c'est-à-dire des mariages entre cousins (*cf.* pedigree de la figure 7.4, les individus I et J se marient mais ont un ancêtre en commun B, c'est un cas de consanguinité, le cycle est mis en évidence en gras), apparaissent des boucles par alliance (*cf.* pedigree de la figure 7.5 aucun mariage n'est fait entre individus ayant un ancêtre commun, mais il existe un cycle alternant sommets *individu* et sommets *mariage*, il est présenté en gras sur la figure 7.5). Ces types de pedigree sont fréquement présents dans les populations d'élevage comme les bovins par exemple.

Les pedigrees dits de demi-frères, sont des pedigrees sans boucles sur une génération. La figure 7.6 en est un exemple.

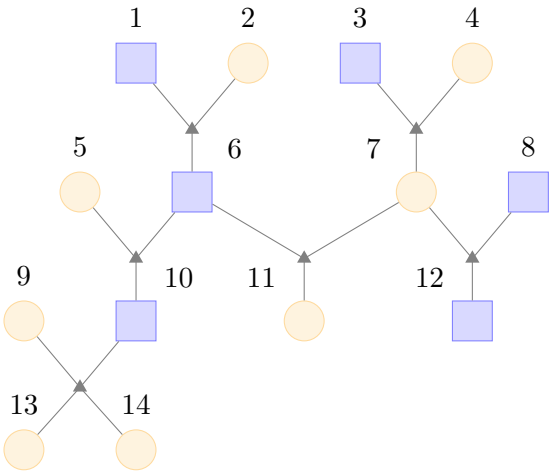


Figure 7.3 – Un pedigree de 14 individus. Les individus 2, 4, 5, 7, 9, 11, 13 et 14 sont des femelles, les autres des mâles. Les fondateurs de ce pedigree sont les individus 1, 2, 3, 4, 5, 8 et 9. Les individus 9 et 10 sont les parents des individus 13 et 14, ils sont reliés par un sommet de mariage.

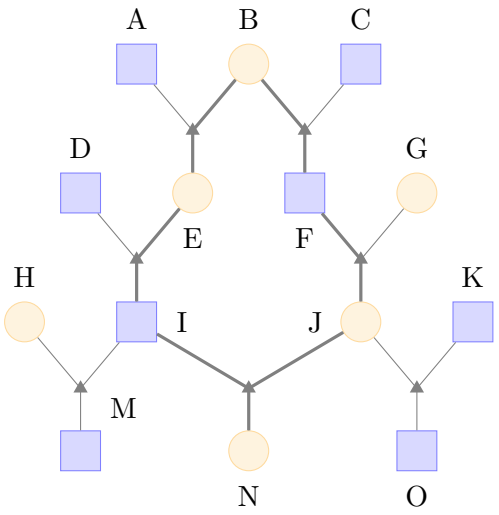


Figure 7.4 – Pedigree avec boucle de consanguinité. Les individus I et J ont un même ancêtre commun B, et ils sont parents de l'individu N.

7.2 Différents cadres d'étude

7.2.1 Méthodes en population

Il existe beaucoup de méthodes de reconstruction d'haplotypes à partir d'individus non apparentés. Mais elles peuvent se regrouper en quatre familles de méthodes : méthode de Clark, les algorithmes de coalescences, l'approche de parcimonie pure et les méthodes statistiques.

La méthode de Clark [Clark, 1990] est le premier algorithme, pour les haplotypes, basé sur les

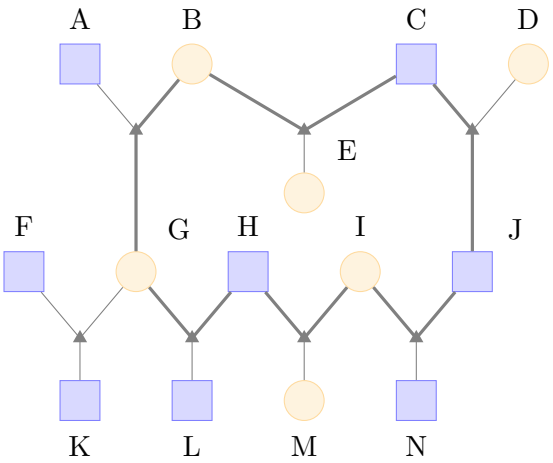


Figure 7.5 – Pedigree complexe avec boucle par alliance

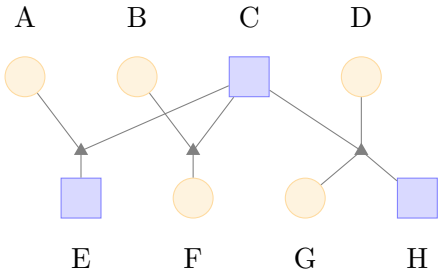


Figure 7.6 – Pedigree de demi-frères

données de génotypes. L'algorithme est le suivant :

Initialisation Les génotypes ne pouvant être construits qu'à partir d'une unique paire d'haplotypes : les génotypes homozygotes à tous les loci ou hétérozygotes sur un unique locus. Ces haplotypes sont ajoutés à un ensemble $\mathcal{H}$ d'haplotypes solutions.

Itération Leurs génotypes pouvant être construits à partir d'un haplotype présent dans $\mathcal{H}$. Si le second haplotype n'est pas dans $\mathcal{H}$, il est ajouté.

Finalisation Aucun génotype ne peut plus être construit à partir des haplotypes de $\mathcal{H}$.

Les méthodes de coalescence supposent pour une population que tous les haplotypes sont issus d'un unique ancêtre, ainsi ils ont tendance à se regrouper en fonction des mutations qui sont apparues. Dans ce cadre, un algorithme dit de Phylogénie Parfaite (PPH pour Perfect PHylogeny) proposé par [Gusfield, 2002] permet de déduire les haplotypes à partir d'une reconstruction phylogénétique.

Le but de certaines méthodes statistiques est d'estimer la distribution (les fréquences) des haplotypes et d'associer avec la plus grande probabilité, une paire d'haplotypes compatibles pour chaque génotype. Il y a les méthodes basées sur l'algorithme EM (Expectation Maximization) comme celles présentées dans [Excoffier and Slatkin, 1995, Hawley and Kidd, 1995]. Pour d'autres méthodes uti-

lisées dans PHASE [Stephens and Donelly, 2000] ou BEAGLE [Browning and Browning, 2007] des hypothèses supplémentaires ou un apriori sur la distribution sont faits pour guider la reconstruction des haplotypes.

Pour plus de détails sur les méthodes en population [Niu, 2004].

7.2.2 Méthodes combinatoires utilisant les liens de parenté

Les individus peuvent être très liés les uns aux autres et l'information de pedigree devient une information importante. De nombreuses méthodes ont été proposées pour la résolution du problème de reconstruction d'haplotypes dans les pedigrees. Elles peuvent être classées en deux grandes familles : les approches statistiques (elles seront présentées dans la section 7.3) et les approches combinatoires.

Ces dernières reconstruisent les haplotypes en utilisant les lois d'hérédité de Mendel et optimisent la solution par rapport à différents critères comme minimiser le nombre d'événements de recombinaison ou encore supposer qu'il n'y a pas de d'événement de recombinaison sur les haplotypes.

7.2.2.1 Reconstruction sans événement de recombinaison

Les événements de recombinaison sont rares dans les régions ADN de petites tailles Il est raisonnable dans ces régions de supposer que la reconstruction d'haplotypes ne comporte pas de recombinaison.

Problème Zero Recombinant Haplotype Configuration (ZRHC)

Entrée : Un pedigree et les génotypes des individus du pedigree

Sortie : Les haplotypes compatibles de chaque individu n'ayant aucun événement de recombinaison

Le problème ZRHC fut proposé en premier par Wijsman [Wijsman, 1987], il définit un ensemble d'une vingtaine de règles génétiques logiques permettant de retrouver les haplotypes d'un pedigree. O'Connell [O'Connell, 2000] propose un algorithme basé sur ces règles, qui enregistre les haplotypes d'un ensemble de loci très liés comme allèles d'un seul locus et qui supprime les génotypes incompatibles suivant l'algorithme de Lange et Goradia présenté dans [Lange and Goradia, 1987] et ainsi qui trouve toutes les configurations d'haplotypes compatibles (ayant zéro recombinaison) du pedigree. Les fréquences des haplotypes pour les fondateurs sont estimées à partir d'un algorithme EM [Dempster et al., 1977].

Une extension de cet algorithme est proposée par Zhang et al. [Zhang et al., 2005] (HAPLORE) et est basée sur la généralisation des règles génétiques de Wijsman supposant l'absence de recombinaison.

7.2.2.2 Minimiser le nombre d'événements de recombinaisons

L'hypothèse précédente d'absence de recombinaison peut être contredite lorsque nous avons une carte dense de loci avec une région d'étude plus étendue [Li and Jiang, 2003]. Dans ce cas minimiser le nombre de ces événements devient plus adéquat.

Problème Minimum Recombinant Haplotype Configuration (MRHC)

Entrée : Un pedigree et les génotypes des individus du pedigree

Sortie : Les haplotypes compatibles de chaque individu tels que le nombre de recombinaisons est minimal sur l'ensemble du pedigree

Qian et Beckmann proposent un algorithme [Qian and Beckmann, 2002] basé sur six règles. Il consiste à effectuer une approximation du problème, en effet il trouve les configurations d'haplotypes pour chaque famille nucléaire qui minimisent le nombre de recombinaisons dans chacune d'elles et non sur l'ensemble du pedigree. Cette méthode ne peut être utilisée qu'avec de petites familles avec un petit nombre de loci. Pour pouvoir l'utiliser sur des pedigrees plus importants Li and Jiang ont étendu cette méthode en utilisant la programmation en nombres entiers. Cette méthode est implémentée dans le logiciel PedPhase [Li and Jiang, 2003].

Exemple 7.7 *Soit une famille nucléaire (père-mère-fils) génotypée sur 4 loci :*

Père :	ab	bb	bb	bb
Mère :	ab	ab	aa	ab
Fils :	aa	bb	ab	ab

Deux configurations d'haplotypes possibles minimisant le nombre de recombinaisons :

1. *Père :* $\frac{\text{bbbb}}{\text{abbb}}$, *Mère :* $\frac{\text{abba}}{\textbf{baaa}}$, *Fils :* $\frac{\text{abbb}}{\text{ab}\textbf{aa}}$

2. *Père :* $\frac{\text{bbbb}}{\text{abbb}}$, *Mère :* $\frac{\text{aaba}}{\textbf{bbaa}}$, *Fils :* $\frac{\text{abbb}}{\text{a}\textbf{baa}}$

Le fils a reçu le second haplotype complet de son père, de sa mère il a reçu les allèles mis en gras (dans l'haplotype du fils et de la mère). Dans les deux configurations, il y a eu un événement de recombinaison entre les loci 2 et 3 dans la première solution et entre les loci 1 et 2 dans la seconde. En minimisation du nombre de recombinaisons ces deux solutions sont équivalentes cependant la recombinaison ne se fait pas entre les mêmes loci, en fonction de la distance génétique entre eux le taux de recombinaison ne peut pas être le même.

Par exemple, considérons que la carte génétique suivante (position en cM) :

Locus 1 *0 cM*

Locus 2 *30 cM*

Locus 3 *40 cM*

Les loci 2 et 3 sont plus proches (10 cM) que les loci 1 et 2 (30 cM), il est plus probable que l'événement de recombinaison se soit produit entre les loci 1 et 2.

Minimiser le nombre de recombinaisons nous amène à obtenir plusieurs solutions ayant le même nombre de recombinaison (*cf.* l'exemple 7.7) mais nous n'avons aucun moyen, à ce niveau, de distinguer la qualité de chacune d'elles. Pour cela, il faut évaluer leur vraisemblance, la section suivante détaille un ensemble de méthodes calculant la solution (configuration d'haplotypes compatible) la plus vraisemblable.

7.3 Méthodes statistiques utilisant les liens de parenté

Pour les petits pedigrees où tous les individus sont génotypés sur un faible nombre de loci, il n'est pas difficile d'énumérer toutes les configurations d'haplotypes compatibles, de calculer leur vraisemblance et d'identifier la (les) plus probables [Sobel et al., 1996]. Mais il est impossible de le faire lorsque les pedigrees sont plus grands avec plus de loci et des génotypes manquants.

7.3.1 Méthodes itératives

Lors de l'analyse de liaison, seuls certains loci sont utilisables pour étudier la transmission des allèles d'un parent à ses descendants. Ces loci sont dits *informatifs* ils permettent d'étudier les recombinaisons lors de la méïose chez les parents. Ainsi pour reconstruire la phase (les deux haplotypes) des parents, il suffit de s'intéresser aux loci informatifs de chaque descendant.

Définition 7.8 Locus informatif
Pour un individu, un locus est *informatif* si un de ses parents est hétérozygote et qu'il est homozygote à ce locus. Son *locus informatif voisin à gauche* est le dernier locus informatif le précédant le long du chromosome. Son *locus informatif voisin à droite* est le premier informatif le suivant. Deux loci informatifs voisins forment une paire informative pour l'individu.

Exemple 7.9 *Soit un père et un de ses fils avec les génotypes suivants sur 7 loci*

Génotypes du père :	ab	bb	aa	ab	ab	aa	ab
Génotypes du fils :	bb	ab	aa	aa	bb	ab	bb

Les loci informatifs pour le fils 1 sont 1, 4, 5 et 7
Les paire informatives sont (1,4) (4,5) (5,7). Ainsi le locus 4 admet (1,4) comme paire informative à gauche et (4,5) à droite. Par contre le locus 1 n'admet pas de paire informative à gauche.

Windig et Meuwissen proposent dans [Windig and Meuwissen, 2004] d'étendre la méthode de Wijsman [Wijsman, 1987] pour prendre en compte l'information de ces loci informatifs. Ils définissent entre deux loci hétérozygotes chez les parents un intervalle.

Pour évaluer les haplotypes parentaux sur ces intervalles plutôt que sur l'ensemble, qui deviendrait beaucoup trop important lorsque le nombre de loci devient grand, ils proposent de calculer la probabilité qu'il soit préférable de modifier les haplotypes à partir d'un des deux loci formant l'intervalle.

Pour un intervalle donné cette probabilité se calcule avec la formule suivante :

$$\mathbb{P}_{switch} = \frac{\prod_{i=1}^{y} r_i \cdot \prod_{j=1}^{x}(1-r_j)}{\prod_{i=1}^{y} r_i \cdot \prod_{j=1}^{x}(1-r_j) + \prod_{i=1}^{y}(1-r_i) \cdot \prod_{j=1}^{x} r_j}$$

où r_i représente le taux de recombinaison entre les deux loci informatifs pour le descendant considéré, y le nombre de descendants ayant une recombinaison entre leurs loci informatifs considéré par rapport à la phase actuelle et x le nombre de descendants n'ayant pas de recombinaisons entre ces loci.

Ainsi l'algorithme est le suivant :

1. Trouver les paires de loci informatifs pour chaque descendant et déterminer le taux de recombinaison correspondant à partir de la carte génétique.
2. Calculer pour chaque intervalle de loci $\mathbb{P}_{switch}$ par rapport à la phase actuelle du parent.
3. Si la probabilité de plusieurs intervalles est supérieure à 0.5, choisir celui dont la probabilité est la plus grande et échanger l'allèle paternel avec l'allèle maternel pour tous les loci à gauche (ou à droite) de l'intervalle considéré en incluant le locus de gauche (ou de droite) de l'intervalle.
4. Répéter ces étapes tant qu'une probabilité $\mathbb{P}_{switch}$ est $\geqslant 0.5$

Exemple 7.10 *Cet exemple est extrait de [Windig and Meuwissen, 2004].*
Soit un père et ses trois fils, seuls les loci hétérozygotes chez le père sont conservés :

Père :	ab	ab	ab	ab
Fils 1 :	aa	aa	aa	bb
Fils 2 :	ab	bb	ab	aa
Fils 3 :	aa	ab	bb	bb
Fils 4 :	aa	ab	ab	aa

Les distances entre les loci sont supposées être identiques (10 cM) avec comme taux de recombinaison 9.5, 18.2 et 25.9% pour des intervalles respectivement de 1, 2 et 3 loci.

	Itération 1				*Itération 2*				*Itération 3*			
Haplotypes	a	a	a	a	a	a	a	b	b	a	a	b
parentaux	b	b	b	b	b	b	b	a	a	b	b	a
Allèles informatifs chez												
le fils 1	a	a	a ·	b	a	a	a	b	a ·	a	a	b
le fils 2	–	b ·	– ·	a	–	b	–	a	–	b	–	a
le fils 3	a ·	– ·	b	b	a ·	– ·	b ·	b	a	–	b ·	b
le fils 4	a	–	–	a	a ·	– ·	– ·	a	a	–	–	a
$\mathbb{P}_{switch}$	*0.142*	*0.429*	***0.613***		***0.576***	*0.231*	*0.387*		*0.424*	*0.002*	*0.072*	

Les tirets '-' représentent les loci non informatifs, les points '·' représentent les événements (possibles) de recombinaison par rapport aux haplotypes parentaux courants.

Nous détaillons le calcul pour le dernier intervalle (entre les loci 3 et 4) dans la première itération. Les taux de recombinaison pour cet intervalle sont 0.095, 0.181, 0.095 et 0.259 (respectivement pour les fils 1, 2, 3 et 4). Il faut calculer la probabilité que la phase parentale change. Si actuellement il n'y a pas de recombinaison entre les deux loci informatifs, le fait que la phase change entraîne l'apparition d'un événement de recombinaison. Ainsi les probabilités chez chaque fils que la phase change sont 0.905, 0.819, 0.095 et 0.259. Toutes ces valeurs sont multipliées (0.0182), on calcule l'événement inverse (que la phase ne change pas) (0.0115). Avec ces deux valeurs nous avons $\mathbb{P}_{switch} = \dfrac{0.0182}{0.0182 + 0.0115} = 0.613$ *De la même manière les probabilités des autres intervalles sont calculées. Pour l'itération 1 la troisième valeur et la seule > 0.5, l'algorithme inverse les allèles du locus 4 (ou des loci 1, 2 et 3) dans les haplotypes parentaux. Les haplotypes parentaux passent de* $\dfrac{\text{aaaa}}{\text{bbbb}}$ *à* $\dfrac{\text{aaab}}{\text{bbba}}$.

Le même processus est répété avec ces nouveaux haplotypes parentaux. Dans cet exemple il faut 3 itérations pour ne plus obtenir de probabilité $\mathbb{P}_{switch} > 0.5$

Druet et Georges proposent dans [Druet and Georges, 2010] une méthode nommée LINKPHASE basée sur cet algorithme. Ils ont modifié la probabilité $\mathbb{P}_{switch}$ permettant de simplement inverser la phase pour un locus et non pour tout un fragment. En effet ils calculent la probabilité qu'un allèle soit sur un des deux haplotypes $\mathbb{P}^k_{gauche} = \prod_{i=1}^{y} r_i \cdot \prod_{j=1}^{x}(1 - r_j)$ où k est l'haplotype ($k = 1$ ou 2), r_i représente le taux de recombinaison entre le locus considéré et son voisin informatif à gauche, c'est le même principe pour $\mathbb{P}^k_{droite}$, pour obtenir $P^k = \mathbb{P}^k_{gauche}\ \mathbb{P}^k_{droite}$.

1. Calculer $L^k = \dfrac{P^k}{P^1 + P^2}$
2. Si $L^k > 0.95$ l'allèle testé est affecté à l'haplotype k, sinon il est sur l'autre $(3 - k)$.

Pour les loci non informatifs chez la descendance, LINKPHASE détermine leur origine en fonction des loci déjà affectés et des haplotypes parentaux [Qian and Beckmann, 2002]. Pour chaque paire de loci informatifs, s'ils ont comme origine le même haplotype h parental, c'est-à-dire que l'on observe aucune recombinaison entre eux, alors lorsque la probabilité d'avoir une double recombinaison est < 0.05, LINKPHASE affecte tous les loci du descendant avec ceux présents sur h entre les deux loci informatifs considérés.

7.3.2 Méthodes basées sur les chaînes de Markov

Lander et Green sont les premiers à proposer un algorithme basé sur les chaînes de Markov [Lander and Green, 1987].

Définition 7.11 vecteur d'hérédité
Le vecteur d'hérédité $\mathbf{v}_l = (p_1, m_1, \ldots, p_n, m_n)$ décrit le résultat des méioses paternelles et maternelles

des n individus non fondateurs du pedigree étudié au locus l.
$p_i = 0$ (resp. 1) si l'allèle grand-paternel (resp. grand-maternel) a été transmis au i^e non fondateur par son père ; m_i représente la même information mais pour la transmission maternelle.

Des représentations similaires ont été proposées dans le contexte de l'analyse de liaison avec une méthode de Monte-Carlo [Sobel and Lange, 1993, Thompson, 1994].

Lander et Green considèrent le HMM avec les variables cachées $\mathbf{v}_l$ et les variables observables $\mathbf{M}_l$ où $\mathbf{M}_l$ représentent les génotypes observés au locus l pour tous les individus (non fondateurs) du pedigree. L'espace des états cachés est l'ensemble Ω des valeurs possibles du vecteur d'hérédité $\mathbf{v}_l$. Les probabilités de transition entre les états dépendent des recombinaisons présentes entre le locus l et le locus $l+1$ (pour l'ensemble des individus).

Le logiciel GENEHUNTER [Kruglyak et al., 1996] utilise ce HMM décrit, pour reconstruire les configurations d'haplotypes pour L loci en utilisant deux méthodes, une exacte et une approchée. La méthode approchée permet de sélectionner le vecteur d'hérédité le plus probable pour chaque locus, avec l'algorithme forward-backward [Rabiner, 1989]. La méthode exacte utilise l'algorithme de Viterbi pour traiter tous les loci en même temps et sélectionner l'ensemble des vecteurs d'hérédité tel que la probabilité jointe $\mathbb{P}(\mathbf{v}_1, \ldots, \mathbf{v}_L \mid \mathbf{M}_1, \ldots, \mathbf{M}_L)$ soit la plus grande.

En temps et en mémoire, GENEHUNTER est linéaire par rapport au nombre de loci mais exponentiel par rapport au nombre d'individus non fondateurs dans le pedigree. Ce qui lui permet de traiter des pedigrees avec beaucoup de loci mais un petit nombre ($\leqslant 15$) d'individus.

Dans le logiciel MERLIN [Abecasis et al., 2002], les auteurs utilisent des "sparse gene flow tree" pour pouvoir utiliser l'algorithme de Viterbi [Rabiner, 1989] et l'algorithme de Lander et Green avec des données plus conséquentes. Ces arbres de flots sont une représentation compacte des arbres binaires complets qui regroupe "gene flow pattern" ayant des résultats identiques dans les feuilles de l'arbre qui sont symétriques ou sans solution.

7.3.3 Méthodes basées sur les réseaux bayésiens

7.3.3.1 Réseaux bayésiens pour l'analyse de liaison : Réseaux de ségrégation

Le pedigree est à la base du problème d'analyse de liaison. Il définit une probabilité jointe sur les génotypes et les observations des individus présents dans le pedigree.

La construction de ce réseau bayésien est proposée dans [Friedman et al., 2000, Lauritzen and Sheehan, 2003] avec trois types de variables aléatoires :

Alléliques Pour chaque individu i, au locus l nous définissons les variables aléatoires A^P_{il} et A^M_{il}. Elles prennent comme valeur l'allèle transmis respectivement par le père et la mère de l'individu i.

Variable de Ségrégation Elles sont similaires à l'approche proposée par Lander et Green. S^P_{il} et S^M_{il} correspondent respectivement à l'origine (paternelle ou maternelle) de l'allèle transmis par

le père et la mère de l'individu i. Formellement, soit p le père de i

$$A_{il}^P = \begin{cases} A_{pl}^P & \text{si } S_{il}^P = P \\ A_{pl}^M & \text{si } S_{il}^P = M \end{cases}$$

A_{il}^M est défini de manière équivalente.

Observées Pour chaque individu i, au locus l nous avons une variable aléatoire G_l^i qui représente l'observation du génotype au locus l de l'individu i.

La figure 7.8 présente un extrait de réseau bayésien illustrant les interactions parents-enfants sur trois loci. Chaque table de probabilités dans le réseau bayésien correspond à une des catégories suivantes :

Modèle de transmission : $\mathbb{P}(A_{il}^P \mid A_{pl}^P, A_{pl}^M, S_{il}^P)$ et $\mathbb{P}(A_{il}^M \mid A_{ml}^P, A_{ml}^M, S_{il}^M)$, où p et m sont les parents de l'individu i.

Modèle de pénétrance : $\mathbb{P}(G_l^i \mid A_{il}^P, A_{il}^M)$

Modèle de recombinaison : $\mathbb{P}(S_{i1}^P) = \mathbb{P}(S_{i1}^M) = \frac{1}{2}$ $\mathbb{P}(S_{il}^P \mid S_{il-1}^P)$ et $\mathbb{P}(S_{il}^M \mid S_{il-1}^M)$ ces deux probabilités conditionnelles admettent la même table.

Fréquence allélique de la population : $\mathbb{P}(A_{il}^P)$ et $\mathbb{P}(A_{il}^M)$ lorsque i est fondateur.

Les tables de probabilités conditionnelles de chacune des catégories sont données à la figure 7.7. Pour une famille nucléaire (père-mère-fils) et trois loci, le réseau bayésien de ségrégation associé est donnée par la figure 7.8.

S_{il}^P		P				M			
A_{pl}^P		a		b		a		b	
A_{pl}^M		a	b	a	b	a	b	a	b
A_{il}^P	a	1	1	0	0	1	0	0	1
	b	0	0	1	1	0	1	1	0

(a) Modèle de transmission

A_{il}^P		a		b	
A_{il}^M		a	b	a	b
G_i^P	aa	1	0	0	0
	ab	0	1	1	0
	bb	0	0	0	1

(b) Modèle de pénétrance

S_{il}^P \ $S_{i(l-1)}^P$	P	M
P	$(1 - r_{(l-1)l})$	$r_{(l-1)l}$
M	$r_{(l-1)l}$	$(1 - r_{(l-1)l})$

(c) Modèle de recombinaison

Figure 7.7 – Les tables de probabilités du réseau pour les variables paternelles, ce sont les mêmes pour les variables maternelles

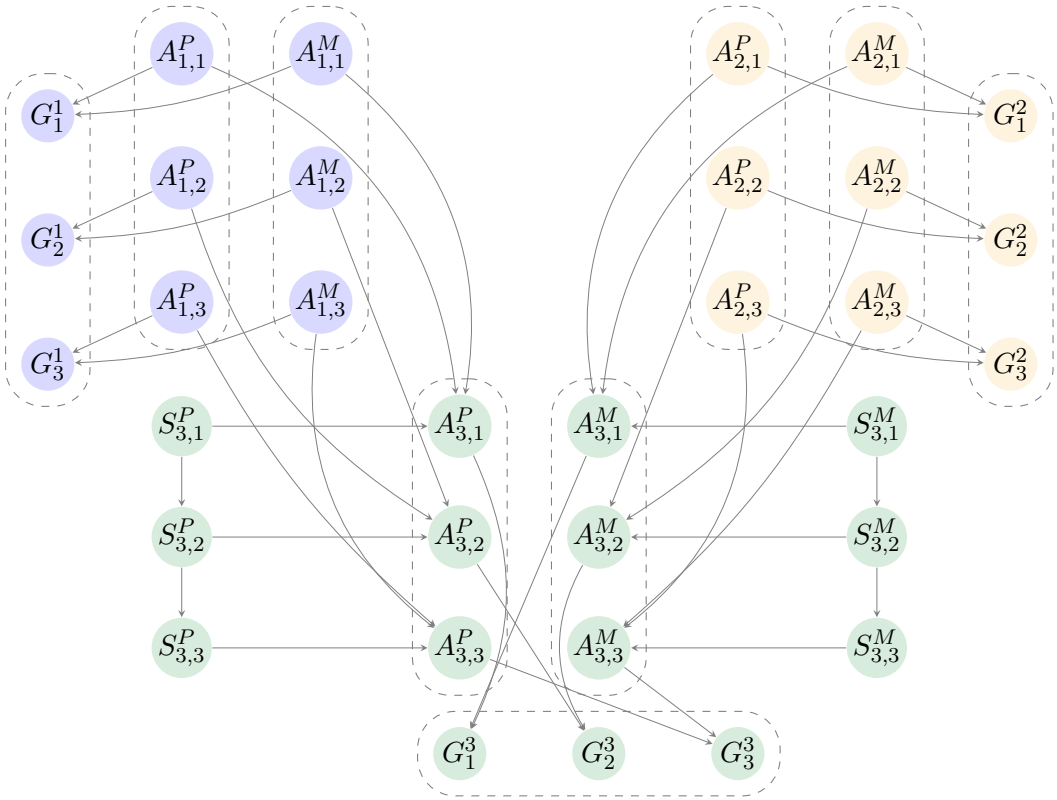


Figure 7.8 – Extrait d'un réseau bayésien, représentant les interactions entre les parents et leur descendance sur 3 loci

7.3.3.2 Algorithme

A partir de cette modélisation en réseau bayésien, Fichelson *et al.* [Fichelson et al., 2005] combinent l'algorithme Elim-Mpe (élimination de variables pour Mpe) avec du conditionnement dans le logiciel Superlink. L'algorithme se déroule de la manière suivante :

1. Elimination de génotype : en pré-traitement, l'algorithme de Lange et Goradia est utilisé pour éliminer les valeurs de génotypes incompatibles.
2. Allele Recording : cette étape permet de fusionner plusieurs allèles d'un même locus, lorsqu'ils n'apparaissent pas dans les observations du pedigree étudié. La fréquence allélique de ce "nouvel allèle" est la somme des fréquences alléliques de tous ceux qu'il résume.
3. Elim-Mpe : pour finir Superlink utilise l'algorithme Elim-Mpe pour obtenir la configuration la plus probable.

Superlink peut ainsi résoudre des pedigrees avec une centaine d'individus et un petit nombre de loci, de petits pedigrees avec une centaine de loci, ou des pedigrees avec un nombre moyen d'individus et de loci.

7.4 Conclusion

Le problème de reconstruction d'haplotypes est essentiel pour la recherche de maladies, pour expliquer des caractères particuliers (la détection de QTL [4]). Depuis une vingtaine d'années ce problème est étudié, de nombreuses méthodes ont été proposées dans le cadre d'étude des populations et lorsque les individus sont apparentés.

Lorsque les individus sont regroupés en familles la reconstruction peut se faire sans recombinaison lorsque la région d'ADN est de petite taille. Pour des régions plus grandes et des cartes génétiques denses, la minimisation du nombre de recombinaisons a été étudiée. Mais ces méthodes ne prennent pas en compte la distance génétique qui existe entre les loci, ce que font les méthodes comme Merlin et Superlink qui utilisent des chaînes de Markov ou des réseaux bayésiens pour évaluer la configuration la plus probable.

La plus grande limite de toutes ces méthodes est de ne pouvoir résoudre à la fois pour un pedigree de grande taille, avec des généalogies complexes, et un nombre conséquent de loci. La taille de données des pedigrees animaux étant vraiment considérable, la contribution principale (*cf.* chapitre 8) de cette thèse est d'étudier les pedigrees à généalogie simple que sont les pedigrees de demi-frères.

4. Quantitative Trait Loci

Une formulation directe pour les pedigrees de demi-frères 8

Ce chapitre se place dans le cadre particulier des pedigrees de demi-frères. Ces pedigrees sont très fréquents dans les populations d'animaux de rente. La sélection des individus reproducteurs se fait quasi exclusivement sur les individus mâles. Ce chapitre décrit une nouvelle formulation compacte de la reconstruction d'haplotype lorsque l'on s'intéresse exclusivement aux haplotypes des reproducteurs, cependant conditionnellement à ces haplotypes, la reconstruction de ceux de la descendance est facile à réaliser [Druet and Georges, 2010].

Nous considérons un pedigree de demi-frères constitué d'un père $\boldsymbol{p}$ et n de ses enfants (fils et filles) supposés indépendants (de mères différentes). Nous avons les génotypes sur L loci (situés sur un même chromosome) pour le père et ses n descendants. Nous supposerons également qu'il n'y ait pas d'erreur mendelienne [1] sur les génotypes, et nous ferons les hypothèses essentielles suivantes :

1. Les crossing-over sont indépendants (non interférence des crossing-over).
2. Les allèles sont tous équiprobables à tous les loci étudiés.
3. Tous les loci sont en équilibre de liaison.

Dans un premier temps il sera question de réduire le modèle de ségrégation présenté dans le chapitre précendent pour reconstruire les haplotypes du père. Ensuite une nouvelle formulation directe équivalente à la réduction du modèle de ségrégation sera présentée. La modélisation sous forme quadratique a été initialement proposée par Jean-Michel Elsen [2].

8.1 Une réduction du modèle de ségrégation

Pour réaliser la réduction du modèle de ségrégation dans le cas de la reconstruction d'haplotypes du père, nous avons besoin des notations et opérations élémentaires qui suivent.

1. Une erreur mendélienne est le fait que les génotypes observés ne respectent pas les lois de Mendel. En effet des erreurs de mesure sont possibles, il existe des méthodes pour "corriger" ces erreurs de mesure. Ici nous supposons que les données que nous avons ont été corrigées.
2. Inra Saga, Toulouse.

Notation 8.1 Soient $\boldsymbol{p}$ le père, $\mathbf{V}$ l'ensemble des variables du modèle de ségrégation et $\mathbf{O}$ l'ensemble des variables d'observation.

- $\mathcal{P}$ l'ensemble des variables de $\boldsymbol{p}$
- $\mathcal{F}_i$ l'ensemble des variables du descendant i
- $\mathcal{M}_i$ l'ensemble des variables de la mère de i
- $\mathcal{A}_i^P$ (resp. $\mathcal{A}_i^M$) l'ensemble des variables "allèle paternel" (resp. marternel) de i
- $\mathcal{S}_i^P$ (resp. $\mathcal{S}_i^M$) l'ensemble des variables de ségrégration paternelle (resp. maternelle) de i
- $\mathcal{H}_i$ l'ensemble des loci hétérozygotes *ordonnés* de i ;
  ainsi $\mathcal{H}_i = \{k_1, k_2, \ldots, k_{|H_x|}\}$ tel que $\forall u < v,\ k_u < k_v$
- $\mathbb{C}^i$ l'ensemble des paires (k, l) de loci consécutifs dans l'ensemble $\mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$

Définition 8.2 opérations élémentaires
Soient X et Y deux variables.

`elim(X)` opération d'élimination de la variable X (*cf.* définition 3.18 page 36 du chapitre 3).

`fus(X,Y)` opération de fusion de deux variables, mise à jour des probabilités existantes sur X et Y (*cf.* section 3.2.4.2 page 49 du chapitre 3).

`dec(X,Y)` opération de décomposition par paire d'une fonction de probabilité par rapport à X et Y (*cf.* chapitre 4).

Définition 8.3 Trouver la paire d'haplotypes la plus probable pour le père $\boldsymbol{p}$ revient au problème MAP suivant :

Entrée : L'ensemble $\mathbf{V}$ et les observations $\mathbf{O}$
Sortie : Une affectation de $\mathcal{P}$ maximisant $\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V}, \mathbf{G})$

Nous considérons un pedigree de $2n+1$ individus composé d'un mâle et n femelles fondateurs et de n descendants (1 descendant par femelle).

L'équation (8.1) de la présente page détaille l'expression de $\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V}, \mathbf{G})$ et l'équation (8.2) page ci-contre une réécriture mettant en évidence les parties concernées par la sommation ou non.

$$\begin{aligned}\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V}, \mathbf{G}) \quad = \quad & \sum_{\mathbf{V}\setminus\mathcal{P}} \Bigg(\prod_{k=1}^{|L|} \mathbb{P}(A^P_{\boldsymbol{p}k}) \cdot \mathbb{P}(A^M_{\boldsymbol{p}k}) \cdot \mathbb{P}(G^{\boldsymbol{p}}_k \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}) \\ & \times \prod_{i=1}^{n} \mathbb{P}(A^P_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{\texttt{mere}(i)k}) \\ & \times \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik}) \\ & \times \mathbb{P}(G^i_k \mid A^P_{ik}, A^M_{ik}) \cdot \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \cdot \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)}) \Bigg)\end{aligned} \tag{8.1}$$

$$\begin{aligned}\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V},\mathbf{G}) \quad = \quad & \left(\prod_{k=1}^{|L|} \mathbb{P}(A^P_{\boldsymbol{p}k}) \cdot \mathbb{P}(A^M_{\boldsymbol{p}k}) \cdot \mathbb{P}(G^{\boldsymbol{p}}_k \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k})\right) \\ & \times \sum_{\mathbf{V}\setminus\mathcal{P}} \Big(\prod_{k=1}^{|L|} \prod_{i=1}^{n} \mathbb{P}(A^P_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{\texttt{mere}(i)k}) \\ & \times \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik}) \\ & \times \mathbb{P}(G^i_k \mid A^P_{ik}, A^M_{ik}) \cdot \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \cdot \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)})\Big)\end{aligned} \tag{8.2}$$

8.1.1 Principes de la réduction

Hormis le père, chaque individu n'appartient qu'à une seule famille nucléaire. Chacune de ces familles est ainsi composée d'une mère et d'un descendant.
Soit $\texttt{Fam}_i = \mathcal{M}_i \cup \mathcal{F}_i$ l'ensemble des variables du descendant i et de sa mère m et $\texttt{Fam}_i \cap \texttt{Fam}_j = \emptyset$, $\forall i \neq j$ formalise cette séparation des familles nucléaires.

$$\begin{aligned}\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V},\mathbf{G}) \quad = \quad & \left(\prod_{k=1}^{|L|} \mathbb{P}(A^P_{\boldsymbol{p}k}) \cdot \mathbb{P}(A^M_{\boldsymbol{p}k}) \cdot \mathbb{P}(G^{\boldsymbol{p}}_k \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k})\right) \\ & \times \prod_{i=1}^{n} \sum_{\texttt{Fam}_i} \Big(\prod_{k=1}^{|L|} \mathbb{P}(A^P_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{\texttt{mere}(i)k}) \\ & \times \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik}) \\ & \times \mathbb{P}(G^i_k \mid A^P_{ik}, A^M_{ik}) \cdot \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \cdot \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)})\Big)\end{aligned} \tag{8.3}$$

Chaque paire mère-descendant (m, i) étant indépendante des autres paires mère-descendant (m', j), les calculs effectués sous les hypothèses de travail sur la formule (8.4) de la présente page peuvent être transposés à l'identique sur les F_j pour tous les autres descendants j de $\boldsymbol{p}$.

$$\begin{aligned}F_i = \quad & \sum_{\texttt{Fam}_i} \Big(\prod_{k=1}^{|L|} \mathbb{P}(A^P_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{\texttt{mere}(i)k}) \\ & \times \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik}) \\ & \times \mathbb{P}(G^i_k \mid A^P_{ik}, A^M_{ik}) \cdot \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \cdot \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)})\Big)\end{aligned} \tag{8.4}$$

La figure 8.2 page 147 illustre les grandes étapes de la réduction pour un pedigree composé d'une seule famille nucléaire. Ces étapes et leurs justifications sont les suivantes :

Etape 1 : *Elimination des variables associées à la mère m : $\mathcal{M}_i$*

Ces variables n'apportant pas d'information au réseau bayésien, et étant peu connectées à l'ensemble du réseau leur sommation est facile à réaliser. Pour chacune des variables éliminées il n'y

a qu'une seule fonction de probabilité qui porte sur elles (au moment de leur élimination).

Pour chaque locus k,
- $\texttt{elim}(A^P_{\texttt{mere}(i)k})$
- $\texttt{elim}(A^M_{\texttt{mere}(i)k})$
- $\texttt{dec}(A^M_{ik}, S^M_{ik})$

Etape 2 : *Elimination des variables "maternelles" associées au descendant i*

Après l'étape 1 les variables de ségrégation maternelle $\mathcal{S}^M_i$ forment une chaîne totalement déconnectée du reste du réseau, et n'ayant pas d'observation sur ces variables, leur somme vaut 1. Les variables d'allèles maternels ont un degré de 1 : leur sommation est simple à réaliser.

Pour chaque locus k,
- $\texttt{elim}(S^M_{ik})$
- $\texttt{elim}(A^M_{ik})$

Etape 3 : *Fusion des variables allèles du père* $\boldsymbol{p}$

Le lien, de différence ou d'égalité suivant les cas, entre les variables $A^P_{\boldsymbol{p}k}$ et $A^M_{\boldsymbol{p}k}$ est tel qu'il est intéressant dans certaines configurations des génotypes observés au locus k (père héterozygote au locus) de fusionner ces deux variables en une variable binaire H_k. Dans ce cas les deux variables sont booléennes et associées à une fonction bijective de différence.

Pour chaque locus k hétérozygote chez le père : $H_k = \texttt{fus}(A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k})$

Etape 4 : *Elimination des variables "allèles paternels" associées au descendant i*

Après les précédentes étapes la sommation des variables de $\mathcal{A}^P_i$ est rendue plus simple car leur nombre de voisins à grandement diminué, de plus leur sommation à ce stade permet de garder une certaine logique biologique au réseau. Au travers des fonctions restantes les notions d'héritages sont encore exprimées de manière explicite.

Pour chaque locus k,
- $\texttt{elim}(A^P_{ik})$

$\texttt{elim}(A^P_{ik})$ engendre différentes situations selon les génotypes observés. Lorsque le père est homozygote au locus k, $\texttt{elim}(A^P_{ik})$ est tout simplement une affectation de A^P_{ik}. Lorsque le père et le descendant i sont hétérozygotes au locus k, $\texttt{elim}(A^P_{ik})$ crée une fonction constante entre H_k et S^P_{ik}. Dans ce cas il n'existe pas réellement de lien entre H_k et S^P_{ik}, puisque peu importe leur

valeur la fonction est constante. Enfin lorsque le fils est homozygote $\texttt{elim}(A_{ik}^P)$ crée une fonction déterministe entre H_k et S_{ik}^P.

Etape 5 : *Elimination des variables de ségrégation paternelle associées au descendant i*

Cette étape utilise les propriétés de la distance de Haldane pour obtenir les probabilités de recombinaison entre deux loci quelconques et la décomposition par paire. Elle permet à la fin d'obtenir un réseau ne portant plus que sur les variables H_k (anciennement les variables allèles du père $\boldsymbol{p}$), s'apparentant à un problème MAX-2SAT.

Pour chaque locus k,

- $\texttt{elim}(S_{ik}^P)$
- $\texttt{dec}(S_{i(k+1)}^P, H_{k-1})$

Remarque 8.4 *Le calcul de chaque étape est détaillé dans l'annexe A page 181.*

L'ordre des étapes n'est pas totalement figé. En effet l'étape 3 peut être réalisée à tout moment. Cependant pour le détail des calculs j'ai choisi de la positionner en troisième.

A l'issue des 5 étapes $\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V},\mathbf{G})$ est un produit de fonctions des variables H_k créées tel que :

$$\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V},\mathbf{G}) = \quad C \cdot \prod_{i=1}^{n} \prod_{(k,l)\in\mathbb{C}^i} f_{G_k^i,G_l^i}(H_k,H_l) \tag{8.5}$$

avec

$$f_{G_k^i,G_l^i}(H_k,H_l) = \begin{cases} (1-r_{kl}) & \text{si } (G_k^i = G_l^i \text{ et } H_k = H_l) \text{ ou } (G_k^i \neq G_l^i \text{ et } H_k \neq H_l) \\ \\ r_{kl} & \text{si } (G_k^i = G_l^i \text{ et } (H_k \neq H_l) \text{ ou } (G_l^i \neq G_k^i \text{ et } H_l = H_k) \end{cases} \tag{8.6}$$

Propriété 8.5 *Sous les hypothèses suivantes :*

1. *Les crossing-over sont indépendants (non interférence des crossing-over)*
2. *Les allèles sont tous équiprobables à tous les loci étudiés.*
3. *Tous les loci sont en équilibre de liaison.*

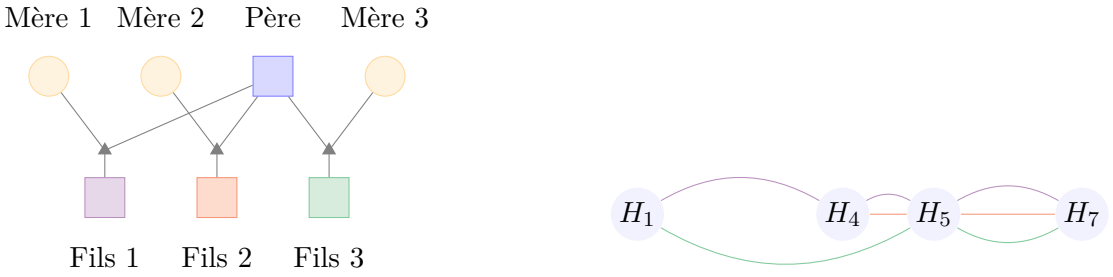
Nous avons $\underset{\mathcal{P}}{\operatorname{argmax}} \Big[\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V},\mathbf{G}) \Big] \equiv \underset{H}{\operatorname{argmax}} \Big[C \cdot \prod_{i=1}^{n} \prod_{(k,l)\in\mathbb{C}^i} f_{G_k^i,G_l^i}(H_k,H_l) \Big]$

Justifications. Les hypothèses permettent de réaliser les 5 étapes et d'obtenir l'équation (8.5) de la présente page la propriété découle directement de cette formule. □

Exemple 8.6 *Considérons un pedigree de* 3 *demi-frères dont les génotypes sont observés sur 7 loci :*

Père :	ab bb aa ab ab aa ab
Fils 1 :	bb ab aa aa bb ab bb
Fils 2 :	ab bb ab aa bb aa aa
Fils 3 :	aa bb aa ab aa ab aa

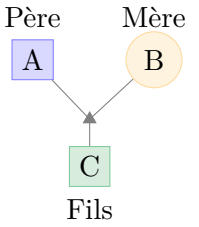
La figure 8.1a représente le pedigree et la figure 8.1b où chaque arête qui compose le graphe est associé à un fils identifié par leur couleur. Le père est hétérozygote aux loci 1, 4, 5 *et* 7, *pour ces loci nous avons une variable* H_l *(étape 3).*
A l'issue des cinq étapes, pour le fils 2 *il existe 2 fonctions, une entre* H_4 *et* H_5 *et l'autre entre les variables* H_5 *et* H_7.



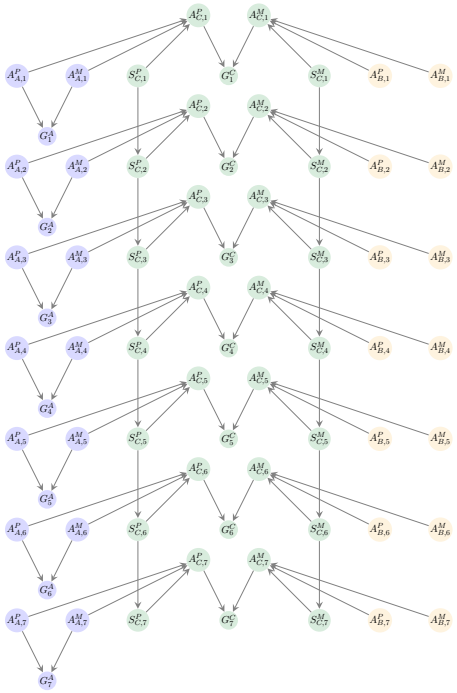
(a) un pedigree de 3 demi-frères

(b) Réseau résultant des 5 étapes

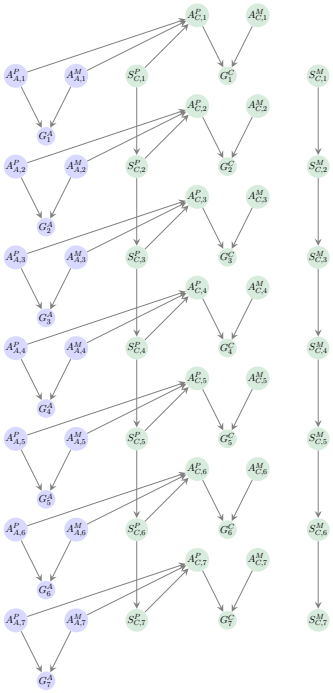
Figure 8.1 – Les génotypes sur 4 loci du père et de ses 3 fils du pedigree en (a) :
Père : $(ab\ bb\ aa\ ab\ ab\ aa\ ab)$ Fils 1 : $(bb\ ab\ aa\ aa\ bb\ ab\ bb)$
Fils 2 : $(ab\ bb\ ab\ aa\ bb\ aa\ aa)$ Fils 3 : $(aa\ bb\ aa\ ab\ aa\ ab\ aa)$



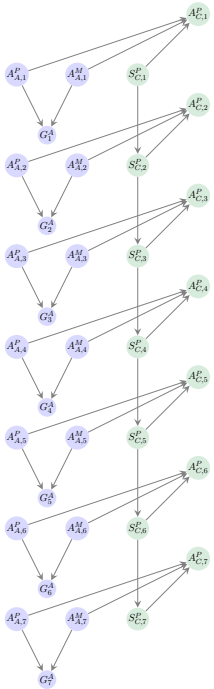
(a) Un pedigree



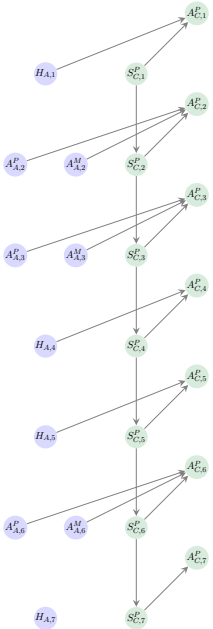
(b) Le réseau de ségrégation



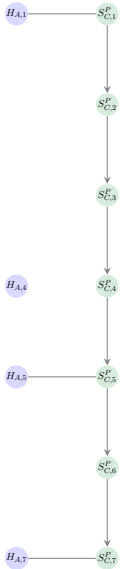
(c) Etape 1



(d) Etape 2



(e) Etape 3



(f) Etape 4

(g) Etape 5

Figure 8.2 – Soit une famille nucléaire en 8.2a. Supposons que le père et le fils soient génotypés sur 7 marqueurs tels que :
Père : $(ab\ bb\ aa\ ab\ ab\ aa\ ab)$. Fils : $(aa\ bb\ aa\ ab\ aa\ ab\ aa)$.
Le réseau d'origine de ségrégation est représenté en 8.2b.
Les réseaux 8.2c à 8.2g sont le résultat graphique des étapes de la réduction du calcul $\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V},\mathbf{G})$

8.2 Une nouvelle formulation directe

Dans cette section, nous considérons seulement les individus génotypés sur les loci de l'ensemble L, ici le mâle et ses n descendants.

Les données du problème, hormis le pedigree, sont les loci, leur génotype pour chaque individu considéré et leur position sur le chromosome.

8.2.1 Les notations

Pour obtenir les deux haplotypes du père étudié, nous avons besoin d'un ensemble de variables permettant de déterminer sur quel chromosome homologue se situe chaque allèle du génotype pour chacun des loci étudiés.

Définition 8.7 vecteur de phasage ou phase
Soit **h** un vecteur de dimension $\mid L \mid$ le *vecteur de phasage* ou *la phase*, tel que pour chaque loci l, le domaine de h_l est $\{-1, 1\}$ définit de la manière suivante :

$$h_l = \begin{cases} 1 & \text{si les allèles sont positionnés tels que } \frac{a}{a}, \frac{b}{b}, \frac{a}{b} \text{ pour les génotypes } aa,bb,ab \\ -1 & \text{si les allèles sont positionnés tels que } \frac{a}{a}, \frac{b}{b}, \frac{b}{a} \text{ pour les génotypes } aa,bb,ab \end{cases}$$

Remarque 8.8 *Pour un locus homozygote l , la valeur de h_l n'a pas d'importance, par la suite elle sera mise par défaut à 1.*

Exemple 8.9 *Supposons que sur 4 loci nous ayons les génotypes suivants pour le père et le vecteur de phase **h** ci-dessous :*

$\mathbf{G}^0$	ab	bb	aa	ab
$\boldsymbol{h}$	-1	1	1	-1

Nous pouvons en déduire que les haplotypes du père sont $\begin{array}{cccc} \text{b} & \text{b} & \text{a} & \text{b} \\ \hline \text{a} & \text{b} & \text{a} & \text{a} \end{array}$

Dans notre modèle, la reconstruction des haplotypes de l'individu choisi (le père dans notre cas) va passer par la phase (ou vecteur de phasage) ayant une probabilité maximale étant donnée les observations de génotypes des descendants. Au lieu d'utiliser directement les génotypes observés dans notre modèle probabiliste nous utilisons une donnée intermédiaire qui est suffisante pour modéliser les événements de la méiose : des vecteurs de transmissions.

Lorsque $\boldsymbol{p}$ est homozygote aa, il est évident qu'il transmet un allèle a, mais il nous est impossible de savoir lequel des deux a été copié et transmis à chaque descendant. Les vecteurs ont pour but d'indiquer quelle copie d'allèle a été transmise aux descendants.

Définition 8.10 valeur et vecteur de transmission
Soit $\mathbf{T}$ une matrice de dimension $(n, \mid L \mid)$, telle que la valeur T_l^i appelée *valeur de transmission* définisse l'origine certaine (de la copie) de l'allèle paternel au locus l du i^{e} fils.

$$T_l^i = \begin{cases} 1 & \text{si le père est hétérozygote, le i}^{\text{e}} \text{ fils homozygote } aa \\ 2 & \text{si le père est hétérozygote, le i}^{\text{e}} \text{ fils homozygote } bb \\ \star & \text{si le père est homozygote ou le i}^{\text{e}} \text{ fils hétérozygote} \end{cases}$$

$\mathbf{T}^i = \begin{pmatrix} T_1^i & \dots & T_{|L|}^i \end{pmatrix}$ est appelé *vecteur de transmission* pour le i^{e} descendant.

Exemple 8.11 *Considérons un père avec 3 fils de mères différentes. Seuls le père et les fils sont génotypés sur 7 loci tels que :*

$\mathbf{G}^0$:	ab	bb	aa	ab	ab	aa	ab
$\mathbf{G}^1$:	bb	ab	aa	aa	bb	ab	bb
$\mathbf{G}^2$:	ab	bb	ab	aa	bb	aa	aa
$\mathbf{G}^3$:	aa	bb	aa	ab	aa	ab	aa

Construction des vecteurs de transmission.

Le père est homozygote aux loci 2,3 et 6 donc pour ces loci la valeur de transmission est $\star$ pour chacun des fils. Pour les autres loci, le père est hétérozygote, il faut donc étudier les génotypes de chaque fils pour compléter leur vecteur de transmission.

Pour le fils 2 : Au locus 1, le fils est hétérozygote comme le père donc on ne peut pas connaître avec certitude lequel des allèles est transmis : $T_1^2 = \star$. Au locus 4, le fils est aa *et le père est* ab*, on peut dire avec certitude le père a transmis le premier allèle : $T_4^2 = 1$. Au locus 5, le fils est homozygote* bb *et le père est* ab*, donc il lui a transmis son second allèle : $T_5^2 = 2$. Pour le locus 7 c'est exactement le même raisonnement.*

Ainsi nous avons la matrice de transmission suivante :

$\boldsymbol{T}^1$:	*2*	$\star$	$\star$	*1*	*2*	$\star$	*2*
$\boldsymbol{T}^2$:	$\star$	$\star$	$\star$	*1*	*2*	$\star$	*1*
$\boldsymbol{T}^3$:	*1*	$\star$	$\star$	$\star$	*1*	$\star$	*1*

La définition suivante est une reformulation de la définition 7.8 page 133 du chapitre précédent.

Définition 8.12 Locus et paire informatifs
Soit $\mathbf{T}^i$ le vecteur de transmission du fils i.

- Un locus l est *informatif* pour i si $T_l^i \neq \star$.
- Une paire (k, l) de loci est dite *informative* si k et l, $(k < l)$, sont deux loci informatifs et qu'il n'existe pas de locus m informatif tel que $k < m < l$.

Exemple 8.13 *Si on poursuit l'exemple précédent nous pouvons déterminer l'ensemble des loci informatifs et les paires informatives pour chaque descendant.*

	loci informatifs	*paires informatives*
descendant 1	*1,4,5,7*	*(1,4) (4,5) (5,7)*
descendant 2	*4,5,7*	*(4,5) (5,7)*
descendant 3	*1,5,7*	*(1,5) (5,7)*

8.2.2 Formulation de la probabilité de la phase sous forme quadratique

Définition 8.14 La probabilité de la phase $\mathbf{h}$ est donnée par

$$\mathbb{P}(\mathbf{h} \mid \mathbf{T}) = \frac{\mathbb{P}(\mathbf{T} \mid \mathbf{h}) \cdot \mathbb{P}(\mathbf{h})}{\sum_{\mathbf{h}'} \mathbb{P}(\mathbf{T} \mid \mathbf{h}') \cdot \mathbb{P}(\mathbf{h}')}$$

L'hypothèse de travail qui est l'équilibre de liaison permet de déduire que $\mathbb{P}(\mathbf{h})$ est une constante quelle que soit la phase $\mathbf{h}$, ainsi nous obtenons $\mathbb{P}(\mathbf{h} \mid \mathbf{T}) = \frac{\mathbb{P}(\mathbf{T} \mid \mathbf{h})}{\sum_{\mathbf{h}'} \mathbb{P}(\mathbf{T} \mid \mathbf{h}')}$. Nous en déduisons que la probabilité de la phase $\mathbf{h}$ est proportionnelle à la vraisemblance des observations ($\mathbb{P}(\mathbf{h} \mid \mathbf{T}) \propto \mathbb{P}(\mathbf{T} \mid \mathbf{h})$). Ainsi la configuration d'haplotypes la plus probable est celle qui maximise $\mathbb{P}(\mathbf{T} \mid \mathbf{h})$.

L'indépendance des événements de méiose produite pour chaque fils [3] justifie l'équation (8.7) et l'équation (8.8) est une simple application de la propriété 2.12 page 20 sur les probabilités conditionnelles.

$$\mathbb{P}(\mathbf{T} \mid \mathbf{h}) = \prod_{i=1}^{n} \mathbb{P}(\mathbf{T}^i \mid \mathbf{h}) \tag{8.7}$$

$$\mathbb{P}(\mathbf{T} \mid \mathbf{h}) = \prod_{i=1}^{n} \mathbb{P}(T_1^i \mid \mathbf{h}) \cdot \mathbb{P}(T_2^i \mid \mathbf{h}, T_1^i) \cdot \mathbb{P}(T_3^i \mid \mathbf{h}, T_1^i, T_2^i) \ldots \mathbb{P}(T_L^i \mid \mathbf{h}, T_1^i, \ldots, T_{L-1}^i) \tag{8.8}$$

Propriété 8.15

$$\mathbb{P}(T_l^i \mid \boldsymbol{h}, T_1^i, \ldots, T_{l-1}^i) = \mathbb{P}(T_l^i \mid h_k, h_l, T_k^i)$$

Justifications. Pour $T_l^i \neq \star$, en faisant les hypothèses de non interférence dans la création des crossing-over et des fréquences alléliques équiprobables à chaque locus, seul le locus informatif courant l et son locus informatif précédent k formant ainsi la paire informative (k, l) sont nécessaires au calcul de $\mathbb{P}(T_l^i \mid \mathbf{h}, T_1^i, \ldots, T_{l-1}^i)$.

En supposant l'indépendance des crossing-over, la probabilité de recombinaison entre k et l ne dépend pas des éventuelles recombinaisons entre les loci 1 et k. De plus chaque valeur de transmission

3. c'est une indépendance conditionnelle. De manière générale, les méioses entre plusieurs enfants sont indépendantes conditionnellement aux génotypes des parents.

valant $\star$ entre les loci $k+1$ et $l-1$ ne modifie pas la vraisemblance puisque les fréquences alléliques sont toutes identiques ainsi la transmission des allèles des mères vers leur fils n'a pas d'importance.

Par conséquent, seules les valeurs de transmission des loci informatifs dans $\mathbf{T}^i$ sont utilisées. □

Propriété 8.16 *Soient $\boldsymbol{h}$ un vecteur de phase, $\boldsymbol{T}^i$ le vecteur de transmission d'un descendant i.*

$$\mathbb{P}(T_l^i \mid \boldsymbol{h}, T_1^i, \ldots, T_{l-1}^i) = \begin{cases} 1 & \text{si } T_l^i = \star \\ \dfrac{1}{2} & \text{si } \nexists k < l, \text{ tel que } (k,l) \textit{ paire informative} \\ r_{kl}^{1-\delta_{kl}^i(\boldsymbol{h})} \cdot (1-r_{kl})^{\delta_{kl}^i(\boldsymbol{h})} & \text{si } (k,l) \textit{ paire informative} \end{cases}$$

$$\textit{avec } \delta_{kl}^i(\boldsymbol{h}) = \frac{1}{4} \cdot \left(\frac{h_l}{2T_l^i - 3} + \frac{h_k}{2T_k^i - 3} \right)^2$$

Justifications. $\mathbb{P}(T_l^i = \star \mid \mathbf{h}, T_1^i, \ldots, T_{l-1}^i) = 1$ car c'est un sytème complet d'événements.
Pour $T_l^i \neq \star$, par les hypothèses 1 (non interférence des crossing-over) et 2 page 141 (équifréquence des allèles) seul le locus informatif k précédent l (k et l forment une paire informative pour le descendant i) est utile donc

$$\mathbb{P}(T_l^i = \star \mid \mathbf{h}, T_1^i, \ldots, T_{l-1}^i) = \mathbb{P}(T_l^i \mid h_k, h_l, T_k^i)$$

Si l est le premier locus informatif ($\nexists k < l$, tel que (k,l) soit une paire informative) il n'y a pas de préférence entre $h_l = \dfrac{a}{b}$ et $h_l = \dfrac{b}{a}$ car tous les loci précédents sont homozygotes.

D'où $\mathbb{P}(T_l^i \mid \mathbf{h}, T_1^i, \ldots, T_{l-1}^i) = \dfrac{1}{2}$.
Dans le cas où il existe une paire informative (k,l) :

$$\mathbb{P}(T_l^i \mid h_k, h_l, T_k^i) = \begin{cases} r_{kl} & \text{si } (h_k = h_l \text{ et } T_k^i \neq T_l^i) \text{ ou } (h_k \neq h_l \text{ et } T_k^i = T_l^i) \\ 1 - r_{kl} & \text{sinon} \end{cases} \tag{8.9}$$

ou de manière algébrique : $\mathbb{P}(T_l^i \mid h_k, h_l, T_k^i) = r_{kl}^{1-\delta_{kl}^i(\mathbf{h})} \cdot (1-r_{kl})^{\delta_{kl}^i(\mathbf{h})}$ □

Propriété 8.17

$$\log(\mathbb{P}(\boldsymbol{T} \mid \boldsymbol{h})) = \mathcal{K} + {}^t\boldsymbol{h}\boldsymbol{W}\boldsymbol{h}$$

telle que $\mathcal{K}$ soit une constante et $\boldsymbol{W}$ une matrice carrée $\mid L \mid \times \mid L \mid$.

Démonstration.
Nous avons $\mathbb{C}^i$ l'ensemble de ses paires informatives. Tout d'abord nous développons la fonction $\delta_{kl}^i(\mathbf{h})$

qui sera nécessaire par la suite.

$$\begin{aligned}
\delta^i_{kl}(\mathbf{h}) &= \frac{1}{4}\cdot\left(\frac{h_l}{2T^i_l-3}+\frac{h_k}{2T^i_k-3}\right)^2\\
&= \frac{1}{4}\cdot\left(\frac{h_l^2}{(2T^i_l-3)^2}+\frac{h_k^2}{(2T^i_k-3)^2}+2\frac{h_lh_k}{(2T^i_l-3)(2T^i_k-3)}\right)\\
&= \frac{1}{4}\cdot\left(\frac{1}{(2T^i_l-3)^2}+\frac{1}{(2T^i_k-3)^2}+2\frac{h_lh_k}{(2T^i_l-3)(2T^i_k-3)}\right)\\
&= \mathcal{C}+\frac{1}{4}\cdot\left(2\frac{h_lh_k}{(2T^i_l-3)(2T^i_k-3)}\right)\\
&= \mathcal{C}+\frac{h_lh_k}{2(2T^i_l-3)(2T^i_k-3)}\\
&= \frac{1}{2}+\frac{h_lh_k}{2(2T^i_l-3)(2T^i_k-3)}
\end{aligned}$$

$$\mathcal{C}=\frac{1}{4}\cdot\left(\frac{1}{(2T^i_l-3)^2}+\frac{1}{(2T^i_k-3)^2}\right)=\frac{1}{2}\text{ car }T^i_l,T^i_k\in\{1,2\}\text{ donc }(2T^i_l-3)^2=(2T^i_k-3)^2=1$$

$$\begin{aligned}
\log(\mathbb{P}(\mathbf{T}\mid\mathbf{h})) &= \log\left[\prod_{i=1}^n\mathbb{P}(\mathbf{T}^i\mid\mathbf{h})\right]\\
&= \sum_{i=1}^n\log\left[\mathbb{P}(\mathbf{T}^i\mid\mathbf{h})\right]\\
&= \sum_{i=1}^n\log\left[\frac{1}{2}\prod_{(k,l)\in\mathbb{C}^i}\mathbb{P}(T^i_l\mid h_l,h_k,T^i_k)\right]\qquad(\textit{cf.}\text{ propriété 8.16})\\
&= \sum_{i=1}^n\log\left[\frac{1}{2}\prod_{(k,l)\in\mathbb{C}^i}r_{kl}^{1-\delta^i_{kl}(\mathbf{h})}\cdot(1-r_{kl})^{\delta^i_{kl}(\mathbf{h})}\right]\\
&= \sum_{i=1}^n\log\left(\frac{1}{2}\right)+\sum_{(k,l)\in\mathbb{C}^i}\log\left[r_{kl}^{1-\delta^i_{kl}(\mathbf{h})}\cdot(1-r_{kl})^{\delta^i_{kl}(\mathbf{h})}\right]\\
&= n\log\left(\frac{1}{2}\right)+\sum_{i=1}^n\sum_{(k,l)\in\mathbb{C}^i}[1-\delta^i_{kl}(\mathbf{h})]\log(r_{kl})+\delta^i_{kl}(\mathbf{h})\log(1-r_{kl})\\
&= n\log\left(\frac{1}{2}\right)+\sum_{i=1}^n\sum_{(k,l)\in\mathbb{C}^i}\log(r_{kl})+\delta^i_{kl}(\mathbf{h})[\log(1-r_{kl})-\log(r_{kl})]\\
&= n\log\left(\frac{1}{2}\right)+\sum_{i=1}^n\sum_{(k,l)\in\mathbb{C}^i}\log(r_{kl})+\left(\frac{1}{2}+\frac{h_lh_k}{2(2T^i_l-3)(2T^i_k-3)}\right)\log\left(\frac{1-r_{kl}}{r_{kl}}\right)\\
&= \mathcal{K}+\sum_{i=1}^n\sum_{(k,l)\in\mathbb{C}^i}\frac{h_lh_k}{2(2T^i_l-3)(2T^i_k-3)}\log\left(\frac{1-r_{kl}}{r_{kl}}\right)
\end{aligned}$$

$$\mathcal{K}=n\log\left(\frac{1}{2}\right)+\sum_{i=1}^n\sum_{(k,l)\in\mathbb{C}^i}\log(r_{kl})+\frac{1}{2}\log\left(\frac{1-r_{kl}}{r_{kl}}\right)$$

Etant donnés k et l, la somme $\sum_{i\ tq\ (k,l)\in\mathbb{C}^i} \frac{h_l h_k}{2(2T_l^i-3)(2T_k^i-3)}$ permet de considérer tous les fils ayant (k,l) comme paire informative. Cette transformation permet d'exprimer $\log(\mathbb{P}(\mathbf{T} \mid \mathbf{h}))$ comme une double somme sur l et k.

$$\begin{aligned}\log(\mathbb{P}(\mathbf{T} \mid \mathbf{h})) &= \mathcal{K} + \sum_{l=1}^{n}\sum_{k<l} h_l h_k \log\left(\frac{1-r_{kl}}{r_{kl}}\right) \cdot \sum_{i\ tq\ (k,l)\in\mathbb{C}^i} \frac{1}{2(2T_l^i-3)(2T_k^i-3)} \\ &= \mathcal{K} + {}^t\mathbf{h}\mathbf{W}\mathbf{h}\end{aligned}$$

Nous faisons le choix de construire une matrice $\mathbf{W}$ symétrique donc $\forall k,l \in \{1,\dots,L\}$

$$W_{kl} = \begin{cases} \log\left(\frac{1-r_{kl}}{r_{kl}}\right) \cdot \sum_{i\ tq\ (k,l)\in\mathbb{C}^i} \frac{1}{4(2T_l^i-3)(2T_k^i-3)} & \text{si } l \neq k \\ 0 & \text{sinon} \end{cases}$$

□

Propriété 8.18 *Soit la matrice* $\boldsymbol{W}$ *construite dans la preuve de la propriété précédente,*

$$W_{kl} = \frac{1}{4}(N_{kl}^+ - N_{kl}^-)\log\left(\frac{1-r_{kl}}{r_{kl}}\right)$$

telle que N_{kl}^+ *(resp.* N_{kl}^-*) représente le nombre de descendants tel que pour chaque descendant* i $T_k^i = T_l^i$ *(resp.* $T_k^i = T_l^i$*) avec* (k,l) *une paire informative pour ce descendant.*

Justifications. Soit une paire informative (k,l).

Détaillons la somme suivante $\sum_{i\ tq\ (k,l)\in\mathbb{C}^i} \frac{1}{4(2T_l^i-3)(2T_k^i-3)}$

Soit un descendant u tel que $T_k^u = T_l^u$ alors $(2T_l^u-3)(2T_k^u-3) = (2T_l^u-3)^2 = 1$.

Soit un descendant v tel que $T_k^v \neq T_l^v$ alors $(2T_l^v-3)(2T_k^v-3) = 1\cdot(-1) = -1$.

Ainsi

$$\sum_{i\ tq\ (k,l)\in\mathbb{C}^i} \frac{1}{4(2T_l^i-3)(2T_k^i-3)} = \frac{1}{4}\cdot\Big(\underbrace{\sum_{\substack{i\ tq\ (k,l)\in\mathbb{C}^i \\ T_k^i=T_l^i}} 1}_{=N_{kl}^+} - \underbrace{\sum_{\substack{i\ tq\ (k,l)\in\mathbb{C}^i \\ T_k^i\neq T_l^i}} 1}_{=N_{kl}^-}\Big)$$

□

Remarque 8.19 *Le signe de chaque coefficient non nul de la matrice* $\boldsymbol{W}$ *est déterminé par le signe de* $(N_{kl}^+ - N_{kl}^-)$ *car* $\log\left(\frac{1-r_{kl}}{r_{kl}}\right)$ *est toujours positif du fait que* $r_{kl} \leqslant 0.5$.

Exemple 8.20 *Si nous poursuivons l'exemple 8.11 pour lequel nous rappelons la matrice de transmission obtenue :*

$$\begin{array}{lccccccc} \boldsymbol{T}^1 : & 2 & \star & \star & 1 & 2 & \star & 2 \\ \boldsymbol{T}^2 : & \star & \star & \star & 1 & 2 & \star & 1 \\ \boldsymbol{T}^3 : & 1 & \star & \star & \star & 1 & \star & 1 \end{array}$$

Il faut maintenant calculer les valeurs des N^+_{kl} et N^-_{kl} pour chaque paire informative présente chez au moins un descendant.
$N^+_{1,4} = 0$ *;* $N^-_{1,4} = 1$ *car pour le descendant* 1, $T^1_1 \neq T^1_4$ *et (1,4) est une de ses paires informatives*
$N^+_{1,5} = 1$ *car pour le descendant* 3, $T^3_1 = T^3_5$ *et (1,5) est une de ses paires informatives ;* $N^-_{1,5} = 0$.
$N^+_{4,5} = 0$ *;* $N^-_{4,5} = 2$ *car pour les descendants* 1 *et* 2, $T^1_4 \neq T^1_5$ *et* $T^2_4 \neq T^2_5$.
$N^+_{5,7} = 2$ *pour les descendants* 1 *et* 3 *;* $N^-_{5,7} = 1$ *à cause du descendant* 2.
Les autres $N^+_{k,l}, N^-_{k,l}$ *sont tous égaux à zéro.*

8.2.3 Formulation en réseau de fonction de coûts

La matrice **W** est symétrique et ses coefficients peuvent être négatifs, pour pouvoir définir un $\mathbb{R}$-Wcsp à partir de la matrice, il faut effectuer une transformation pour obtenir les fonctions de coûts utilisées pour le $\mathbb{R}$-Wcsp. Ce dernier est défini dans la définition suivante.

Définition 8.21 Soit $\boldsymbol{\mathcal{P}}_W = (H, D, F)$ un $\mathbb{R}$-Wcsp binaire.
- $H = \{h_1, h_2, \ldots, h_L\}$ l'ensemble des valeurs du vecteur de phase pour les L loci étudiés.
- $F = \{f_{kl} \mid W_{kl} \neq 0, k < l\}$ tel que $f(h_k, h_l) = -2 \cdot W_{kl} h_k h_l + \mid W_{kl} \mid$

La table de chaque fonction de coûts de $\boldsymbol{\mathcal{P}}_W$ est une version souple des contraintes de différence ($W_{kl} < 0$) ou d'égalité ($W_{kl} > 0$) :

h_k \ h_l	-1	1
-1	$-4W_{kl}$	0
1	0	$-4W_{kl}$

(a) $W_{kl} < 0$

h_k \ h_l	-1	1
-1	0	$4W_{kl}$
1	$4W_{kl}$	0

(b) $W_{kl} > 0$

Table 8.1 – Tables des fonctions de coûts f_{kl} par rapport au signe de W_{kl}

Elles traduisent le fait que les observations de transmission préconisent que les haplotypes h_k et h_l soient phasés de la même manière (table 8.1b) ou en opposition (table 8.1a)

Propriété 8.22 *Soient le* WCSP $\boldsymbol{\mathcal{P}}_W$ *et* $\mathcal{A}$ *une affectation complète de* $\boldsymbol{\mathcal{P}}_W$*. L'affectation* $\mathcal{A}$ *minimise* $\sum_{f_{kl}\in F} f_{kl}$ *si et seulement si elle maximise* $\log(\mathbb{P}(\boldsymbol{h} \mid \boldsymbol{T}))$.

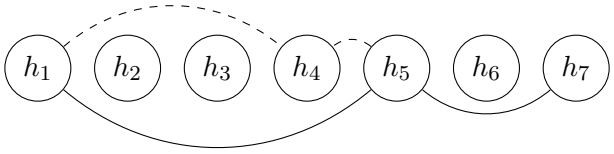
Démonstration. De façon triviale nous avons $\underset{\mathbf{h}}{\operatorname{argmax}} \log(\mathbb{P}(\mathbf{h} \mid \mathbf{T})) = \underset{\mathbf{h}}{\operatorname{argmin}} [- \log(\mathbb{P}(\mathbf{h} \mid \mathbf{T})]$.

$$
\begin{aligned}
-\log(\mathbb{P}(\mathbf{h} \mid \mathbf{T})) &= -\mathcal{K} - {}^t\mathbf{h}\mathbf{W}\mathbf{h} && \text{d'après la propriété 8.17}\\
&= -\mathcal{K} - \sum_{l=1}^{|L|}\sum_{k<l} 2W_{kl}h_k h_l && \text{car } W_{kl} \text{ est symétrique}\\
&= -\mathcal{K} + \sum_{l=1}^{|L|}\sum_{k<l} -2W_{kl}h_k h_l + \mid W_{kl} \mid - \mid W_{kl} \mid\\
&= -\mathcal{K} + \sum_{l=1}^{|L|}\sum_{k<l}(- \mid W_{kl} \mid) + \sum_{l=1}^{|L|}\sum_{k<l} -2W_{kl}h_k h_l + \mid W_{kl} \mid\\
&= \mathcal{K}_1 + \sum_{l=1}^{|L|}\sum_{k<l} -2W_{kl}h_k h_l + \mid W_{kl} \mid\\
&= \mathcal{K}_1 + \sum_{\substack{l,k \text{ tq}\\ W_{kl}\neq 0}} -2W_{kl}h_k h_l + \mid W_{kl} \mid\\
&= \mathcal{K}_1 + \sum_{f_{kl}\in F} f_{kl}(h_k, h_l)
\end{aligned}
$$

avec $\mathcal{K}_1 = -\mathcal{K} + \sum_{l=1}^{|L|}\sum_{k<l}(- \mid W_{kl} \mid)$.

Ainsi la propriété est démontrée. □

Exemple 8.23 *Reprenons l'exemple 8.11. le graphe de contraintes associé au* $\mathbb{R}$-WCSP $\boldsymbol{\mathcal{P}}_W$ *est le suivant.*



Les arêtes en pointillés représentent les cas où le coefficient W_{kl} *est négatif.*

8.2.4 Extension : les mères génotypées

Jusqu'à présent nous faisions l'hypothèse que les mères n'étaient pas génotypées, mais il arrive dans certains cas qu'elles le soient. Il est généralement intéressant de prendre toute l'information donnée. Dans notre cas, la connaissance des génotypes des mères nous permet de combler certaines incertitudes

sur la transmission père-fils. Cette incertitude est levée lorsque le père et le fils sont hétérozygotes et la mère est homozygote.

La définition qui suit reprend la totalité de la définition 8.10 page 149 du vecteur de transmissions en ajoutant cette nouvelle information apportée par le génotype des mères.

Notons $\mathbf{M}^d$ la matrice des génotypes observés des mères et $\mathbf{M}^{di}$ le vecteur des génotypes de la mère du i^{e} descendant.

Définition 8.24 vecteur de transmission
Soit $\mathbf{T}$ une matrice de dimension $(n, \mid L \mid)$, telle que la valeur T_l^i appelée *valeur de transmission* définisse l'origine certaine (de la copie) de l'allèle paternel au locus l du i^{e} fils.

$$T_l^i = \begin{cases} 1 & \text{si (1) le père est hétérozygote, le i}^{\text{e}}\text{ descendant homozygote } aa \\ & \text{ou (2) } \textit{le père et le } i^{e} \textit{ descendant sont hétérozygotes et la mère homozygote } \text{bb} \\ 2 & \text{si (1) le père est hétérozygote, le i}^{\text{e}}\text{ descendant homozygote } bb \\ & \text{ou (2) } \textit{le père et le } i^{e} \textit{ descendant sont hétérozygotes et la mère homozygote } \text{aa} \\ \star & \text{si (1) le père est homozygote ou (2) le père, le i}^{\text{e}}\text{ descendant et la mère sont tous} \\ & \text{hétérozygotes} \end{cases}$$

$\mathbf{T}^i = \begin{pmatrix} T_1^i & \dots & T_{|L|}^i \end{pmatrix}$ est le *vecteur de transmission* pour le i$^{\text{e}}$ descendant.

Exemple 8.25 *Supposons qu'en plus des génotypes du père et des 3 fils présentés dans l'exemple 8.11 nous ajoutons l'information des génotypes des mères des 3 descendants.*

$\boldsymbol{M}^0$:	ab	bb	aa	ab	ab	aa	ab								
$\boldsymbol{M}^1$:	bb	ab	aa	aa	bb	ab	bb	$\boldsymbol{M}^{d1}$:	aa	ab	ab	bb	aa	ab	ab
$\boldsymbol{M}^2$:	ab	bb	ab	aa	bb	aa	aa	$\boldsymbol{M}^{d2}$:	ab	ab	bb	ab	ab	aa	ab
$\boldsymbol{M}^3$:	aa	bb	aa	ab	aa	ab	aa	$\boldsymbol{M}^{d3}$:	aa	ab	ab	aa	ab	ab	aa

$\boldsymbol{T}^1$:	*2*	$\star$	$\star$	*1*	*2*	$\star$	*2*
$\boldsymbol{T}^2$:	$\star$	$\star$	$\star$	*1*	*2*	$\star$	*1*
$\boldsymbol{T}^3$:	*1*	$\star$	$\star$	***2***	*1*	$\star$	*1*

Pour le descendant 2*, au locus* 1 *l'allèle transmis par le père ne peut être identifié car sa mère, son père et lui-même sont hétérozygotes, donc* T_1^2 *est toujours égal à* $\star$*. Pour le descendant* 3 *au locus* 4*, l'allèle transmis par son père peut être identifié : ils sont tous les deux hétérozygotes* ab *(fils) et* ab *(père) et la mère est* aa*, il est donc certain que la mère a transmis un allèle* a *au descendant, le père lui a donc transmis l'allèle* b*(son* 2^{nd} *allèle) ainsi* $T_4^3 = 2$*. Pour cet exemple il n'y a que cette modification par rapport aux vecteurs de transmissions établis dans l'exemple 8.11 page 149. Les* N^+ *et* N^- *sont inchangés sauf pour* $N_{1,4}^-$ *qui vaut* 2 *maintenant par les descendants* 1 *et* 3 *et* $N_{4,5}^- = 3$*.*

Le graphe de contraintes est le même que celui présenté dans l'exemple 8.23 page précédente.

Remarque 8.26 *Nous avons intégré ce modèle dans le logiciel* TOULBAR2 *permettant d'obtenir les haplotypes du père en lisant directement les fichiers de pedigree. Le modèle a également été intégré au logiciel* `QTLMap` [4].

8.3 Equivalence des modèles

Soit un pedigree $\mathscr{P}$ de n demi-frères, dont seuls le père et les descendants sont génotypés sur un ensemble de loci L. Nous avons l'ensemble $\mathbf{V}$ des variables du réseau bayésien modélisant le modèle de ségrégation, l'ensemble $\mathbf{G}$ des observations et la matrice de transmission $\mathbf{T}$.

Propriété 8.27 *L'ensemble (k,l) des loci consécutifs de $\mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$ forment l'ensemble des paires informatives du descendant i.*

Justifications. $\mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$ est l'ensemble des loci l tels que le père soit hétérozygote et le descendant i homozygote, donc $T_l^i \neq \star$. Soit (k,l) une paire de loci consécutifs de $\mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$. Par la définition des loci consécutifs (*cf.* 8.12 page 149), il n'existe pas de locus $m \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$, tel que $k < m < l$ d'où $T_k^i \neq \star$ $T_l^i \neq \star$ et pour tout $k < m < l$, $T_k^i = \star$ donc (k,l) est une paire informative du descendant i. □

Remarque 8.28 *L'ensemble des paires de loci consécutifs de $\mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$ et l'ensemble des paires informatives du descendant i étaient déjà notés par $\mathbb{C}^i$ dans les deux cas. Nous garderons donc cette notation pour ces ensembles (identiques).*

Pour un descendant i et un locus $l \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$, les informations contenues dans G_l^i et T_l^i répresentent exactement la même chose. Pour ce locus l, $T_l^i \neq \star$, donc $T_l^i = 1$ (resp. $T_l^i = 2$) c'est-à-dire que i est homozygote aa *(resp.* bb*) et $G_l^i =$* aa *(resp. $G_l^i =$* bb*).*

De même pour la variable H_l résultant de `fus`$(A^P_{\boldsymbol{pk}}, A^M_{\boldsymbol{pk}})$ *nous avons : $H_l = \dfrac{\text{a}}{\text{b}}$ (resp.$\dfrac{\text{b}}{\text{a}}$) est équivalent à $h_l = 1$ (resp. -1) par leur définition respective.*

Théorème 8.29 *Soient $\mathcal{P} \subset \mathbf{V}$ les variables portant sur le père et $\boldsymbol{h}$ le vecteur de phase de ses haplotypes.*

$$\underset{\mathcal{P}}{\operatorname{argmax}} \sum_{\mathbf{V} \setminus \mathcal{P}} \mathbb{P}(\mathbf{V}, \mathbf{G}) \equiv \underset{\boldsymbol{h}}{\operatorname{argmax}}\, \mathbb{P}(\boldsymbol{T} \mid \boldsymbol{h})$$

Démonstration. Sous les hypothèses de travail posées en début de chapitre et rappelées à la propriété 8.5 page 145 nous pouvons simplifier l'expression de $\sum_{\mathbf{V} \setminus \mathcal{P}} \mathbb{P}(\mathbf{V})$ et obtenir l'équivalence suivante :

$$\underset{\mathcal{P}}{\operatorname{argmax}} \sum_{\mathbf{V} \setminus \mathcal{P}} \mathbb{P}(\mathbf{V}) \equiv \underset{H}{\operatorname{argmax}} \prod_{i=1}^{n} \prod_{(k,l) \in \mathbb{C}^i} f_{G_l^i, G_k^i}(H_l, H_k)$$

4. http://www.inra.fr/qtlmap

avec $H = \{H_1, \ldots, H_{|L|}\}$ l'ensemble des variables H_k résultant de $\texttt{fus}(A^P_{\boldsymbol{pk}}, A^M_{\boldsymbol{pk}})$ et l'équation (8.6) page 145 pour deux loci k et l formant une paire informative dont je rappelle l'expression :

$$f_{G^i_k,G^i_l}(H_k, H_l) = \begin{cases} (1 - r_{kl}) & \text{si } (G^i_k = G^i_l \text{ et } H_k = H_l) \text{ ou } (G^i_k \neq G^i_l \text{ et } H_k \neq H_l) \\ \\ r_{kl} & \text{si } (G^i_k = G^i_l \text{ et } (H_k \neq H_l) \text{ ou } (G^i_l \neq G^i_k \text{ et } H_l = H_k) \end{cases}$$

Dans le nouveau modèle nous avons calculé $\mathbb{P}(\mathbf{T} \mid \mathbf{h}) = \frac{1}{2} n \prod_{i=1}^{n} \prod_{(k,l)\in\mathbb{C}^i} \mathbb{P}(T^i_l \mid h_k, h_l, T^i_k)$

Nous rappelons l'expression $\mathbb{P}(T^i_l \mid h_k, h_l, T^i_k)$ calculée lors de la démonstration de la propriété 8.16 page 151 pour une paire informative (k, l) :

$$\mathbb{P}(T^i_l \mid h_k, h_l, T^i_k) = \begin{cases} (1 - r_{kl}) & \text{si } (T^i_k = T^i_l \text{ et } h_k = h_l) \text{ ou } (T^i_k \neq T^i_l \text{ et } h_k \neq h_l) \\ r_{kl} & \text{si } (T^i_k \neq T^i_l \text{ et } h_k = h_l) \text{ ou } (T^i_k = T^i_l \text{ et } h_k \neq h_l) \end{cases}$$

donc $\mathbb{P}(T^i_l \mid h_k, h_l, T^i_k) = f_{T^i_k,T^i_l}(h_k, h_l)$ et puisque pour les loci k et l forment une paire informative de i nous avons $T^i_k \equiv G^i_k$ et $T^i_l \equiv G^i_l$ et les équivalences suivantes.

$$\begin{aligned}
\operatorname*{argmax}_{H} \prod_{i=1}^{n} \prod_{(k,l)\in\mathbb{C}^i} f_{G^i_l,G^i_k}(H_l, H_k) &\equiv \operatorname*{argmax}_{\mathbf{h}} \prod_{\substack{i \text{ tq} \\ (k,l)\in\mathbb{C}^i}} f_{T^i_k,T^i_l}(h_k, h_l) \\
\operatorname*{argmax}_{H} \prod_{i=1}^{n} \prod_{(k,l)\in\mathbb{C}^i} f_{G^i_l,G^i_k}(H_l, H_k) &\equiv \operatorname*{argmax}_{\mathbf{h}} \frac{1}{2} n \prod_{i=1}^{n} \prod_{(k,l)\in\mathbb{C}^i} \mathbb{P}(T^i_l \mid h_k, h_l, T^i_k) \\
\operatorname*{argmax}_{H} \prod_{i=1}^{n} \prod_{(k,l)\in\mathbb{C}^i} f_{G^i_l,G^i_k}(H_l, H_k) &\equiv \operatorname*{argmax}_{\mathbf{h}} \mathbb{P}(\mathbf{T} \mid \mathbf{h})
\end{aligned}$$

finalement,

$$\operatorname*{argmax}_{\mathcal{P}} \sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V}, \mathbf{G}) \quad \equiv \quad \operatorname*{argmax}_{\mathbf{h}} \mathbb{P}(\mathbf{T} \mid \mathbf{h})$$

Le théorème est démontré. □

8.4 Evaluations expérimentales

Tout d'abord nous décrivons le protocole des simulations commun à toutes les évaluations expérimentales sur simulations. Dans les différentes études réalisées, certaines modifications ont été apportées à ce schéma de simulation (changement de la taille du chromosome, nombre de loci, la variation du nombre de descendants ou encore la présence de déséquilibre de liaison). Pour chaque étude les caractéristiques utilisées seront précisées.

Pour toutes les simulations effectuées (exceptée pour le chromosome X), pour les fondateurs, les loci simulés sont en équilibre de liaison, leur fréquence allélique est calculée à partir d'une distribution

Beta (de paramètres $\alpha = 2, \beta = 2$) car c'est similaire aux fréquences observées dans les cheptels bovins. Les événements de recombinaison sur un chromosome sont simulés à partir de la fonction de Haldane (cf. section 6.3 page 120) Les génotypes sont obtenus par permutations aléatoires des deux allèles à chaque locus pour chaque paire d'haplotypes. Les loci sont équidistants le long du chromosome.

La méthode présentée est implémentée dans ToulBar2. La version 0.9.2 de ToulBar2 a été utilisée pour les résultats présentés. Pour résoudre le Wcsp créé nous avons utilisé principalement DFBB, mais également Btd pour les évaluations de la section 8.4.6 page 165. Nous l'avons comparée avec des méthodes de l'état de l'art présentées dans le chapitre précédent. Nous l'avons d'une part comparée à des méthodes exactes : Superlink [Fichelson et al., 2005] version 1.6, Merlin [Abecasis et al., 2002] version 1.1.2, puis d'autre part à des méthodes approchées : LinkPhase [Druet and Georges, 2010] (les paramètres recommandés par les auteurs ont été utilisés et les loci non reconstruits sont fixés arbitrairement après la résolution), `W&M` [Windig and Meuwissen, 2004] (implémentation personnelle en R). Ensuite nous avons étudié sa performance lorsque les données présentaient un déséquilibre de liaison simulé ou naturel. Enfin nous avons étudié les caractéristiques de la formulation en Wcsp de la méthode.

Ces méthodes ont été comparées en terme d'erreur de switch et de temps. Le temps limite autorisé était de 5 minutes. Les résultats présentés sont des moyennes sur 50 familles pour chaque jeu de paramètres utilisés.

Définition 8.30 switch
Soit deux haplotypes réels d'un individu. Lorsque deux segments d'un même haplotype sont reconstruits sur deux haplotypes différents, on parle de *switch*.

(a) Deux haplotypes réels **(b)** Reconstruction des haplotypes

Figure 8.3 – La reconstruction réalisée présente deux erreurs de switch

8.4.1 Comparaison avec le modèle de ségrégation

Caractéristiques des simulations (sans déséquilibre de liaison) :
Longueur de la région étudiée : 1 Morgan
Nombre de loci : 1 500
Nombre de descendants : varie de 1 à 30 (avec un pas de 1)

La version actuelle de ToulBar2 ne permet pas actuellement d'effectuer $\max_{\mathcal{P}} \Big[\sum_{\mathbf{V} \setminus \mathcal{P}} \mathbb{P}(\mathbf{V}, \mathbf{G}) \Big]$. La comparaison faite ici correspond à $\max_{\mathcal{P}} \Big[\max_{\mathbf{V} \setminus \mathcal{P}} \mathbb{P}(\mathbf{V}, \mathbf{G}) \Big]$ telle que $\max_{\mathbf{V} \setminus \mathcal{P}} \mathbb{P}(\mathbf{V}, \mathbf{G})$ est obtenu

en réalisant les opérations adaptées, présentées dans la section 8.1.1 page 143 (`elim, dec, fus`). $\max_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V},\mathbf{G})$ représente une approximation de $\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V},\mathbf{G})$.

Le graphique 8.4b représente le pourcentage d'erreur de switch fait par chacune des modélisations. La faible différence du pourcentage de switch entre les deux modélisations ($< 0.4\%$) s'explique par l'approximation réalisée pour le modèle de ségrégation. En effet, actuellement il n'est pas encore possible de résoudre un problème MAP avec TOULBAR2, les comparaisons ont été réalisées en résolvant le problème MPE sur le réseau de ségrégation puis la solution étudiée, pour le calcul de switch, est restreinte aux variables utiles. Le graphique 8.4a représente le temps de résolutions pour les deux modélisations. Pour le modèle de ségrégation le temps croît linéairement par rapport à la taille du problème (le nombre de variables augmente avec le nombre d'individus dans le pedigree, en effet nous avons $5 \cdot (2n+1) \cdot \mid L \mid$ variables où n est le nombre de demi-frères).

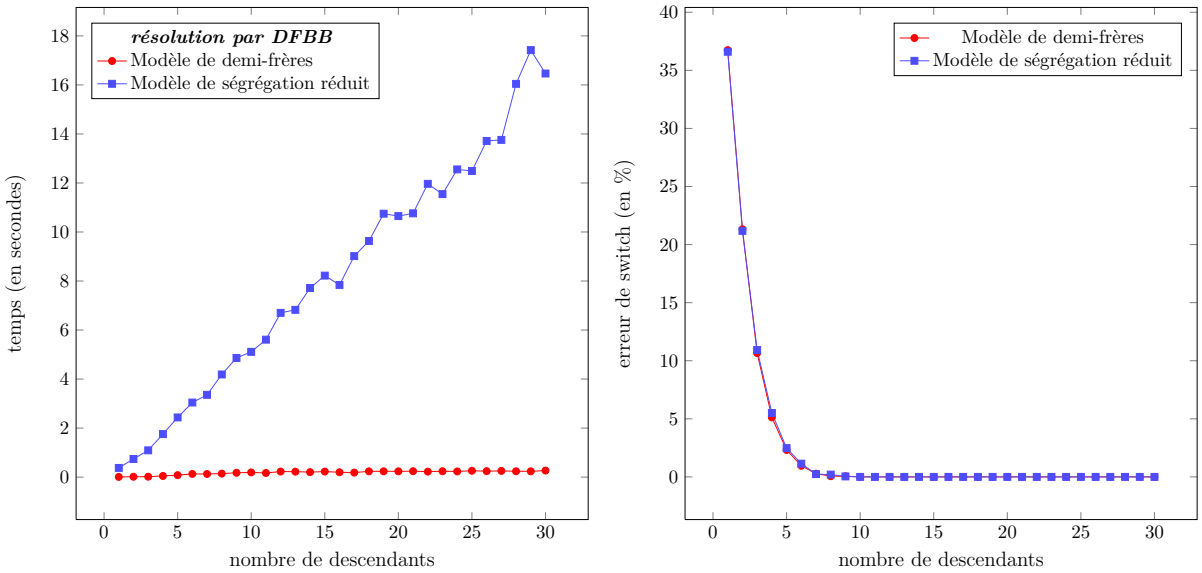


Figure 8.4 – Résultats comparatifs entre le modèle de ségrégation réduit et le modèle direct de demi-frères pour 1 500 loci sur un chromosome de 1 Morgan.

8.4.2 Comparaison avec les méthodes exactes

Caractéristiques des simulations (sans déséquilibre de liaison) :

Longueur de la région étudiée : 1 Morgan
Nombre de loci : 1 500
Nombre de descendants : varie de 1 à 30 (avec un pas de 1)

La figure 8.5 page ci-contre montre les résultats obtenus par notre méthode, MERLIN et SUPERLINK. Dans le cas où toutes les méthodes supposent des fréquences alléliques équiprobables pour tous

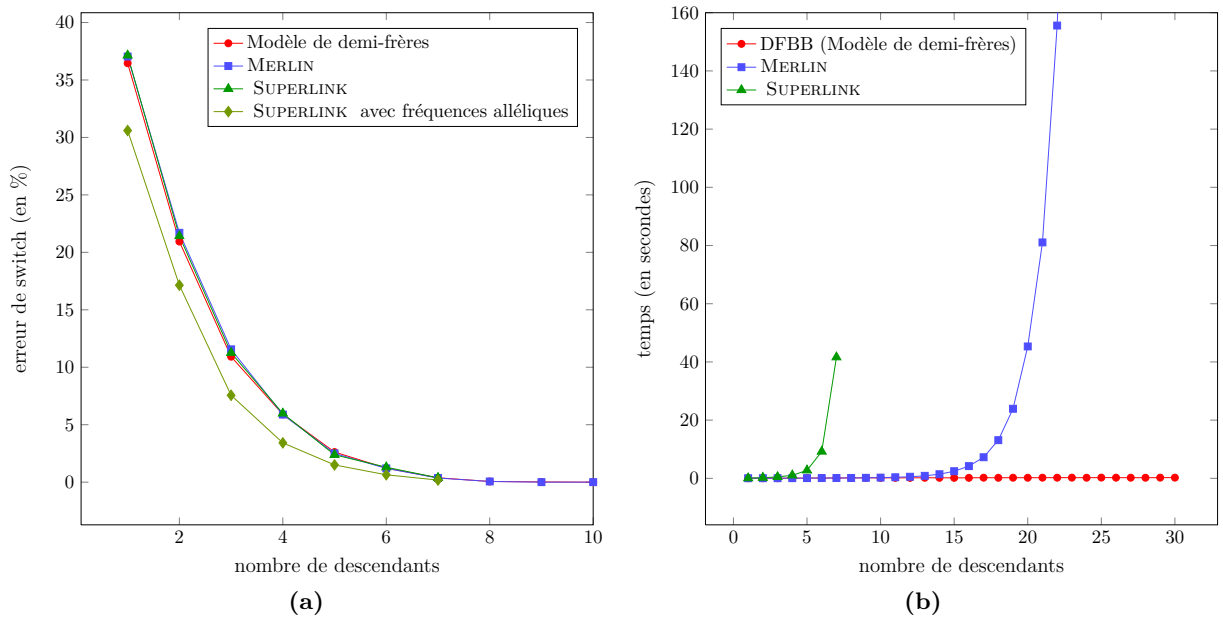


Figure 8.5 – Résultats comparatifs entre notre méthode et les méthodes exactes pour 1 500 loci sur un chromosome d'un Morgan (sans déséquilibre de liaison)

les SNP. Les légères différences d'haplotypes optimaux sont dues au fait que Merlin et Superlink ont reconstruit (et maximisé alors que nous, nous avons intégré) les haplotypes pour l'ensemble des individus. En effet, Merlin reconstruit les haplotypes pour le père et les descendants, Superlink reconstruit en plus ceux pour les mères.

Le pourcentage de switch (figure 8.5a) décroît rapidement avec le nombre de descendants, il est inférieur à 6% (resp. 1%) pour $n = 4$ (resp. 7) descendants. Lorsque Superlink a connaissance des vraies fréquences alléliques, il trouve de meilleurs haplotypes pour le père pour les petites familles. Merlin n'améliore pas ses résultats. Superlink dépasse en mémoire dès qu'il y a plus de 7 descendants (15 individus). Merlin prend plus de 150 secondes pour 22 descendants alors que ToulBar2 utilisant Dfbb met moins d'une seconde (figure 8.5b)

8.4.3 Comparaison avec les méthodes approchées

Caractéristiques des simulations (sans déséquilibre de liaison) :

Longueur de la région étudiée : 1 Morgan

Nombre de loci : 1 500

Nombre de descendants : varie de 4 à 10 (pas de 1) puis de 10 à 100 (pas de 5)

Pour des raisons d'implémentation de la méthode `W&M`, le nombre de descendants ne pouvait pas être inférieur à 4. Pour les familles ayant moins de 10 descendants LinkPhase ne reconstruit pas tous les loci (hétérozygotes), la figure 8.6b page suivante illustre cette tendance, elle représente le

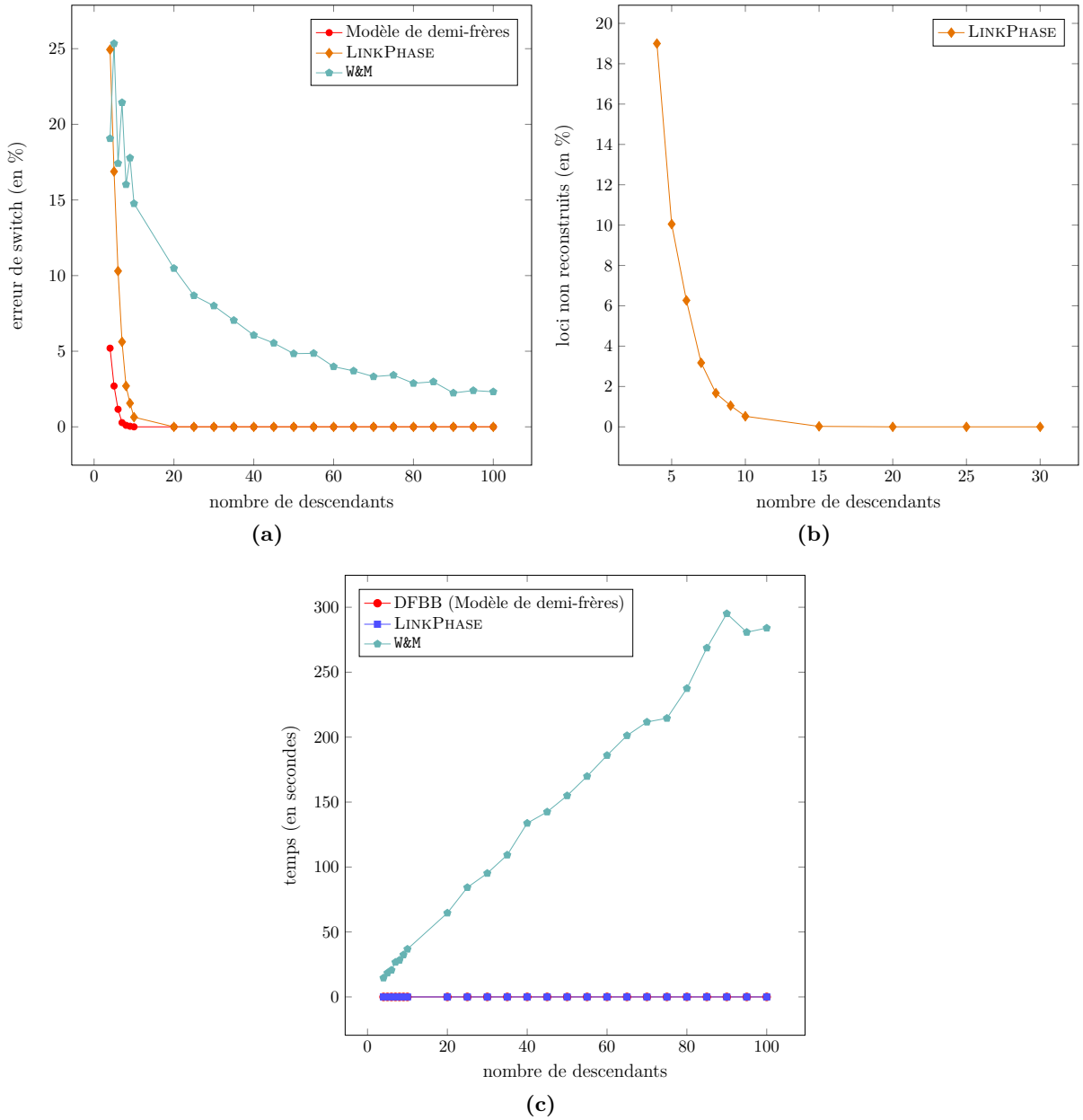


Figure 8.6 – Comparaison avec les méthodes approchée sur un chromosome d'un Morgan et 1 500 loci sans déséquilibre de liaison

pourcentage de loci non reconstruits en fonction de la taille des familles. Pour calculer le nombre de switch, représenté par la figure 8.6a, nous avons rempli aléatoirement les loci non recontruits, ce qui explique un taux d'erreur plus élevé que pour le modèle de demi-frères. Pour les deux méthodes, il n'y a aucun switch lorsque nous considérons seulement les loci reconstruits par LinkPhase. La méthode `W&M` ne converge pas vers les haplotypes réels même avec une centaine de descendants. Alors que pour

20 descendants LinkPhase et notre modèle reconstruisent sans erreur les haplotypes du père, `W&M` reconstruit encore avec 10% de switch, ceci est dû à son étude locale de la reconstruction.

Alors que ToulBar2 et LinkPhase résolvent ces problèmes en moins d'une seconde, le temps d'exécution de `W&M` augmente linéairement par rapport au nombre de descendants (*cf.* figure 8.6c).

8.4.4 Etude du déséquilibre de liaison

Caractéristiques des simulations (avec déséquilibre de liaison simulé) :
Longueur de la région étudiée : 1 Morgan
Nombre de loci : 1 500
Nombre de descendants : varie de 1 à 20 (avec un pas de 1)

Le déséquilibre de liaison a été généré par simulation d'un scénario Wright-Fisher avec cent individus mariés aléatoirement durant cent générations ; les haplotypes mâles et femelles sont sélectionnés dans la dernière génération [5].

Le déséquilibre de liaison est présent dans les populations réelles. Nous avons étudié le comportement de notre modèle de demi-frères ne prenant pas en considération le déséquilibre de liaison sur des données en ayant. La figure 8.7 page suivante représente le pourcentage de switch réalisé par la méthode. Par rapport aux résultats précédents, l'ordre de grandeur est similaire. Il semble même qu'il soit meilleur dans le cas où nous ayons du déséquilibre de liaison (par exemple, pour 2 descendants ici nous avons 17% contre 21% (*cf.* figure 8.5a page 161). Mais le nombre de loci hétérozygotes est plus faible, de l'ordre de 40% de moins, lorsqu'il y a du déséquilibre de liaison, ce qui engendre un nombre d'haplotypes possibles (pour le père et les mères) plus faible également.

8.4.5 Etude de la largeur d'arbre de notre formulation en réseau de fonctions de coûts

Caractéristiques des simulations (sans déséquilibre de liaison) :
Longueur de la région étudiée : 2 Morgan
Nombre de loci : varie de 100 à 10 000 (avec un pas de 500)
Nombre de descendants : 20, 100, 1 000

Pour étudier la difficulté des problèmes résultants en Wcsp, j'ai calculé la largeur d'arbre de leur graphe de contraintes. La figure 8.8 page suivante montre la largeur d'arbre moyenne obtenue par un ordre d'élimination de variables suivant l'ordre des loci le long du chromosome. Elle reste relativement petite, la largeur d'arbre maximale rencontrée dans toutes ces simulations fut 30 (pour 1 000 descendants et 10 000 loci) . Elle semble croître logarithmiquement avec le nombre de descendants, mais compte tenu de la nature du problème de reconstruction d'haplotypes et des pedigrees pouvant être étudiés, les instances à 1 000 descendants sont bien au-delà de la réalité.

5. comme précédemment les fondateurs (de la première génération) sont en équilibre de liaison.

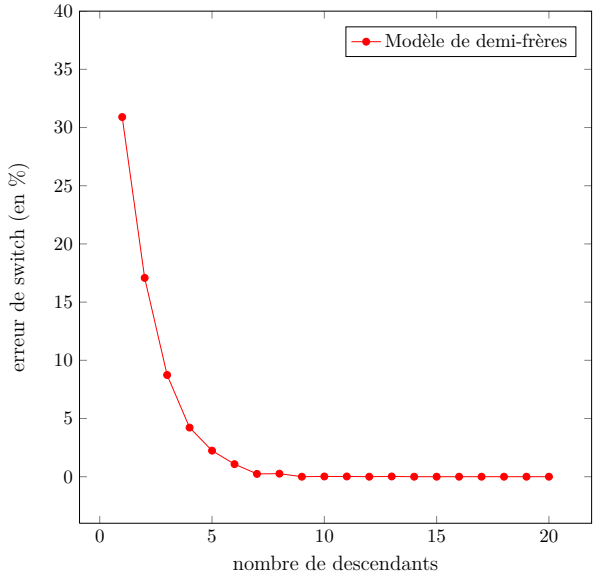


Figure 8.7 – Etude de l'impact du déséquilibre de liaison sur le modèle de demi-frères

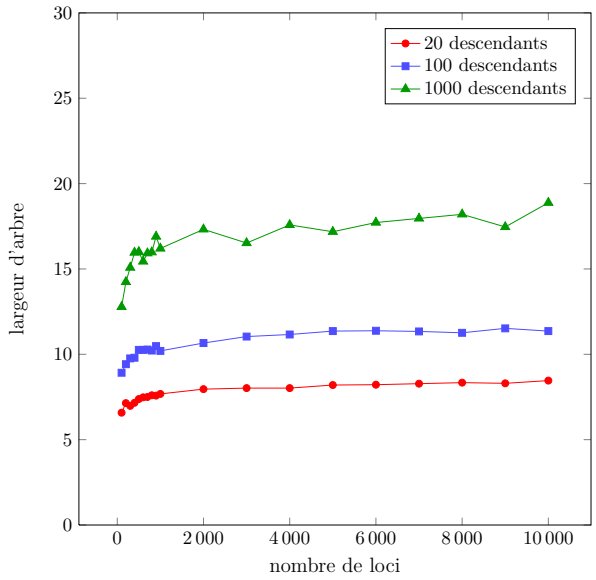


Figure 8.8 – Etude de la largeur d'arbre pour la formulation en WCSP

La largeur d'arbre est suffisamment petite pour ces instances WCSP, elle sont faciles à résoudre avec un algorithme de programmation dynamique. Ainsi pour les jeux de données plus importants suivants nous avons utilisé `VE` et BTD plutôt que DFBB.

8.4.6 Chromosome X humain

Pour cette section, les caractéristiques suivantes sont valables pour les deux résultats présentés (sans et avec les mères génotypées).

Caractéristiques des simulations (avec déséquilibre de liaison naturel) :

Longueur de la région étudiée : 1.64 Morgan

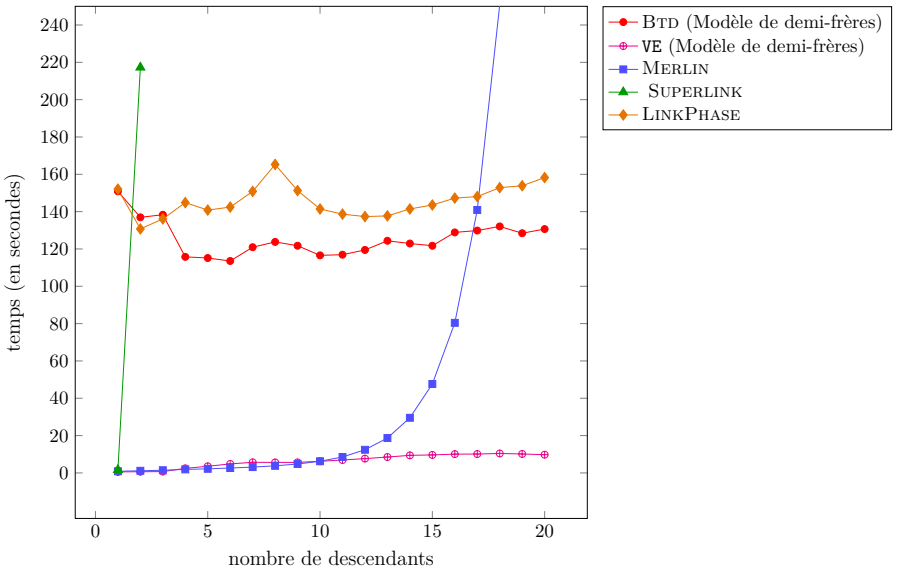
Nombre de loci : 36 000

Nombre de descendants : varie de 1 à 15 (avec un pas de 1)

Pour ces données nous avons pris 44 haplotypes réels du chromosome X des pères de 44 trios de la population CEU (voir le projet HAPMAP phase 3 release 2 sur `www.hapmap.org`) comme haplotypes initiaux pour les fondateurs de la population simulée. Comme les individus mâles n'ont qu'une copie du chromosome X, l'haplotype de ce chromosome est connu avec certitude. Ces données présentent un déséquilibre de liaison naturel. J'ai sélectionné 36 000 SNP, sur une distance de 1.64 Morgan.

8.4.6.1 Mères non génotypées

Pour cette expérimentation nous avons ajouté l'algorithme `VE` [Dechter, 1999] implémenté dans `toolbar` version 3.1 [6]. Les pourcentages de switch sont similaires à ceux des expérimentations précédentes. Pour ce qui est du temps, `VE` est plus rapide mais demande plus d'espace que BTD. De plus BTD prend beaucoup de temps (bien plus de 50%) dans la création de la décomposition arborescente. Lorsque celle-ci est créée la résolution par BTD est très rapide.



(a)

Figure 8.9 – Temps de calcul des méthodes sur le chromosome X humain de 1.64 Morgan et 36 000 loci avec déséquilibre de liaison naturel

6. `carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro`

8.4.6.2 Mères génotypées

L'information des génotypes des mères permet une nette amélioration de la reconstruction des haplotypes du père pour les petites familles de 1 à 4 descendants. Pour une famille avec un seul descendant la connaissance de la mère, permet d'améliorer de plus de 50% la reconstruction des haplotypes paternels. Il suffit de 3 descendants et leur mère pour obtenir une solution optimale avec moins de 1% de switch sur les 11 750 loci hétérozygotes en moyenne.

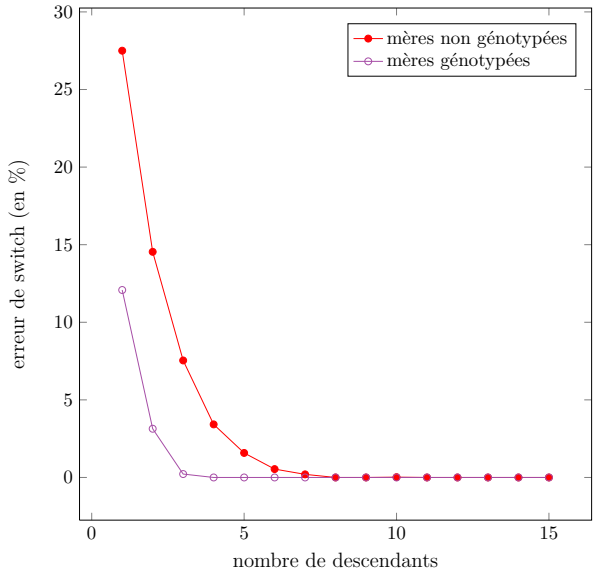


Figure 8.10 – Comparaison du modèle de demi-frères lorsque les mères sont génotypées ou pas

8.5 Une mesure de confiance de la reconstruction

Les familles de petites tailles, de 1 à 4 descendants, posent le problème de la fiabilité de la reconstruction sur certains loci. Dans cette section, il s'agit de présenter une mesure qui évalue la fiabilité (la confiance) de la reconstruction à chaque locus.

8.5.1 Définitions

La meilleure information de fiabilité pour chaque locus est la probabilité a posteriori de chaque allèle.

Définition 8.31 probabilité a posteriori
Soient $\mathbf{H}$ le vecteur de phases, $\mathbf{T}$ le vecteur de transmission (les observations) et un locus l. La probabilité a posteriori pour $h_l = 1$ est :

$$\mathbb{P}(h_l = 1 \mid \mathbf{T}) = \frac{\mathbb{P}(\mathbf{T} \mid h_l = 1)}{\mathbb{P}(\mathbf{T} \mid h_l = 1) + \mathbb{P}(\mathbf{T} \mid h_l = -1)}$$

Définition 8.32 probabilité de confiance
Soient $\mathbf{H}$ le vecteur de phases, $\mathbf{T}$ le vecteur de transmission (les observations), $\mathcal{A} = \{a_1, \ldots, a_L\}$ l'affectation optimale de $\mathbf{H}$ et un locus l. La mesure de confiance $\mathcal{C}(a_l)$ est

$$\mathcal{C}(a_l) = \frac{\mathbb{P}(\mathbf{T}, \mathbf{H}_l = \mathcal{A}_l \mid h_l = a_l)}{\mathbb{P}(\mathbf{T}, \mathbf{H}_l = \mathcal{A}_l \mid h_l = 1) + \mathbb{P}(\mathbf{T}, \mathbf{H}_l = \mathcal{A}_l \mid h_l = -1)}$$

où $\mathbf{H}_l = \mathbf{H} \setminus \{h_l\}$ et $\mathcal{A}_l = \mathcal{A} \setminus \{a_l\}$

Cette mesure est une approximation de la probabilité a posteriori.

8.5.2 Evaluation théorique

Pour tester la qualité de notre mesure de confiance, nous pouvons faire une régression linéaire.

Définition 8.33 régression linéaire
Etant donné un échantillon aléatoire $(Y_i, X_i)_{\{i=1,\ldots,n\}}$ un modèle de régression linéaire suppose une relation affine entre Y_i et X_i :

$$Y_i = aX_i + b$$

Définition 8.34 prédicteur de H
Soit $\mathbf{H}$ notre vecteur de phase et nos observations $\mathbf{T}$. Le prédicteur $\widehat{\mathbf{H}}$ de $\mathbf{H}$ est défini par $\widehat{\mathbf{H}} = E(\mathbf{H} \mid \mathbf{T})$.

Propriété 8.35 *Le prédicteur $\widehat{\mathbf{H}}$ minimise l'erreur quadratique moyenne et n'a pas de biais c'est-à-dire $E_{\mathbf{H}}(\widehat{\mathbf{H}}) = \mathbf{H}$. La régression de $\mathbf{H}$ sur son prédicteur $\widehat{\mathbf{H}}$ nous donne $\mathbf{H} = \widehat{\mathbf{H}} + \varepsilon$*

Justifications. Il est possible de montrer que $\mathrm{Cov}(\widehat{\mathbf{H}}, \varepsilon) = 0$ [Casella and Berger, 2001] donc $\mathrm{Var}(\mathbf{H}) = \mathrm{Var}(\widehat{\mathbf{H}}) + \mathrm{Var}(\varepsilon)$.
Ainsi $\mathrm{Cov}(\mathbf{H}, \widehat{\mathbf{H}}) = \mathrm{Cov}(\widehat{\mathbf{H}} + \varepsilon, \widehat{\mathbf{H}}) = \mathrm{Cov}(\widehat{\mathbf{H}}, \widehat{\mathbf{H}}) + \mathrm{Cov}(\varepsilon, \widehat{\mathbf{H}}) = \mathrm{Cov}(\widehat{\mathbf{H}}, \widehat{\mathbf{H}}) = \mathrm{Var}(\widehat{\mathbf{H}})$.
Par la régression linéaire nous avons $\mathbf{H} = a\widehat{\mathbf{H}} + b$ avec $a = \dfrac{\mathrm{Cov}(\mathbf{H}, \widehat{\mathbf{H}})}{\mathrm{Var}(\widehat{\mathbf{H}})} = \dfrac{\mathrm{Var}(\widehat{\mathbf{H}})}{\mathrm{Var}(\widehat{\mathbf{H}})} = 1$ □

Remarque 8.36 *Il est possible de forcer la régression à passer par l'ordonnée à l'origine ($b = 0$), dans ce cas la valeur de a nous indique si nous avons une sur-estimation ou une sous-estimation de la réalité.*

8.5.3 Evaluation par simulations

Pour évaluer cette nouvelle mesure de confiance, nous avons simulé comme dans la section précédente plusieurs tailles de familles de demi-frères sans les mères génotypées. Une famille de grande taille est générée, à partir de cette famille nous créons 50 familles réduites à un petit nombre de descendants pris aléatoirement, par construction chaque famille est composée du même père. Les trois premiers groupes ne présentent pas de déséquilibre de liaison.

Groupe A	0.835877
Groupe B	1.00238
Groupe C	0.994817
Groupe X	0.9958714

Table 8.2 – Calcul du coefficient a de la régression linéaire

Groupe A : A partir d'une famille de 80 descendants génotypés sur 1 000 loci pour un chromosome d'un Morgan de long, l'évaluation porte sur des familles de 2 descendants.

Groupe B : A partir d'une famille de 80 descendants génotypés sur 1 000 loci pour un chromosome d'un Morgan de long, l'évaluation porte sur des familles de 3 descendants.

Groupe C : A partir d'une famille de 80 descendants génotypés sur 1 000 loci pour un chromosome d'un Morgan de long, l'évaluation porte sur des familles de 4 descendants.

Groupe X : A partir d'une famille de 500 descendants génotypés sur les 36 000 loci du chromosome X présenté dans la section précédente.

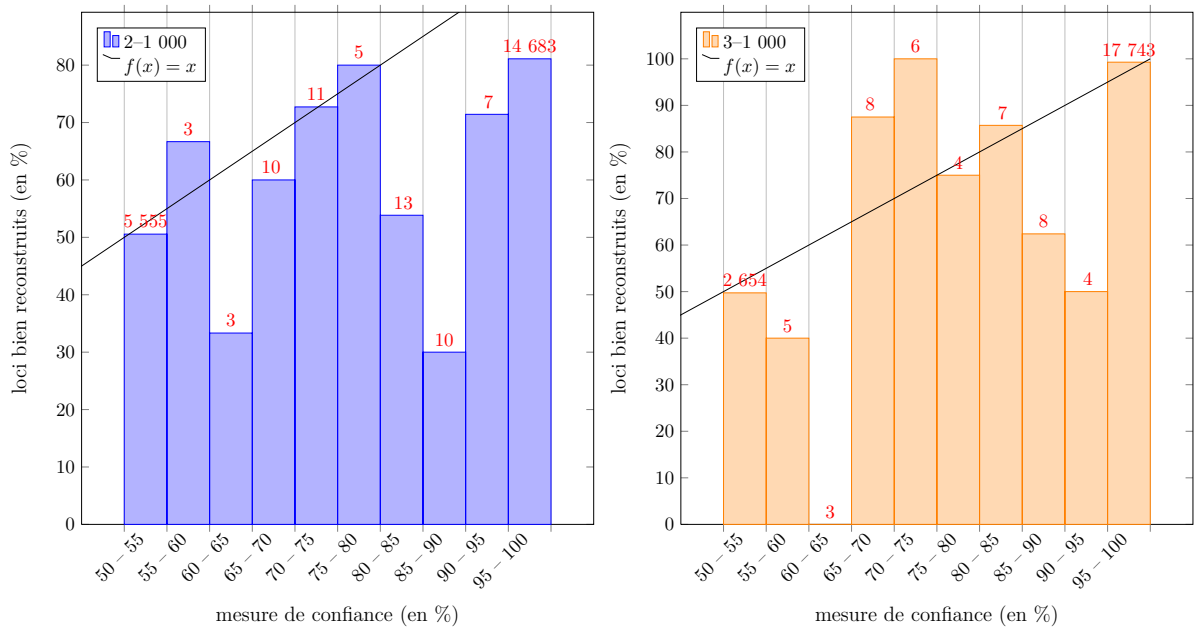
La table 8.2 reporte pour chaque groupe le calcul du coefficient a de la régression linéaire avec $b = 0$. Pour les quatres groupes, ce coefficient est significativement proche de 1. Ce bon résultat nous valide l'utilisation de la mesure dans son ensemble, les graphiques de la figure 8.11 reportent les résultats détaillés pour les différents groupes.

Chaque histogramme reporte le nombre de loci bien reconstruits par rapport à la valeur de leur mesure de confiance $\mathcal{C}$. Pour chaque famille simulée dans les groupes A,B,C et X, nous mesurons $\mathcal{C}(a_l)$. D'après le graphique 8.11d, pour les familles du groupe X, 1 038 loci reconstruits ont leur mesure de confiance comprise entre 90 et 95%, ainsi théroriquement 90% à 95% de ces loci ont bien été reconstruits, ici le pourcentage réel obtenu est de 98.46 ce qui est conforme à la théorie.

Pour les groupes A,B,C, les familles sont de petites tailles en termes de loci, les résultats obtenus (*cf.* figure 8.11a,) nous montrent que notre mesure est plutôt optimiste sur la reconstruction, en effet la théorie prédit un taux plus important de reconstruction correcte que la réalité, mais les calculs se font sur un petit nombre de données.

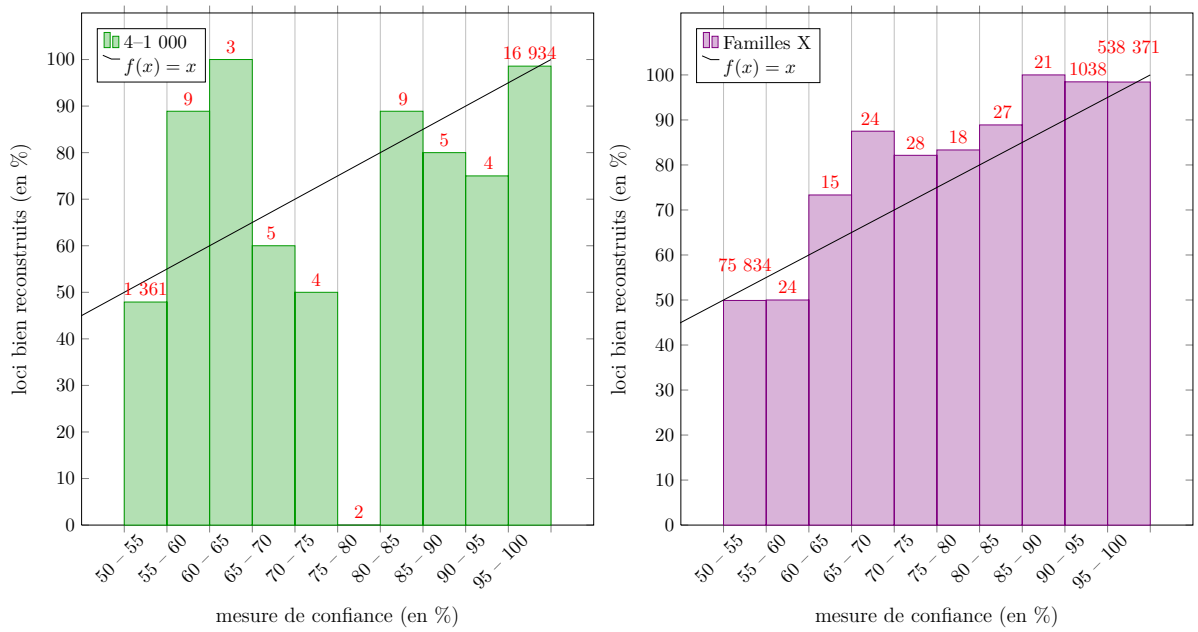
Pour les familles ayant peu de loci pris en compte, la répartition dans les classes de pourcentages nous donne des classes avec un nombre de locus trop faible pour pouvoir émettre une hypothèse concrète sur la qualité de cette mesure dans ces classes. Seules deux classes critiques (50-55% et 95-100%) ont un nombre suffisant d'observation pour tirer la conclusion que cette mesure approxime, pour ces classes, la probabilité a posteriori de chaque locus.

Pour le groupe X (graphique 8.11d), les observations par classes (surtout les intermédiaires) sont un peu plus conséquentes, et nous pouvons remarquer une meilleure régularité dans la comparaison entre la mesure et les observations. Elle confirme l'observation précédente, notre mesure approxime correctement, en moyenne évidemment, la probabilité a posteriori des loci.



(a) Familles : 2 descendants, 1 000 loci

(b) Familles : 3 descendants, 1 000 loci



(c) Familles : 4 descendants, 1 000 loci

(d) Famille X : 3 descendants, 36 000 loci

Figure 8.11 – Histrogrammes représentant le pourcentage de loci bien reconstruits en fonction de notre mesure de confiance. En rouge est indiqué le nombre de loci concernés dans le calcul.

8.6 Conclusion

Ce chapitre a illustré dans une première partie comment le réseau de ségrégation dans le cas de demi-frères pouvait être simplifié, réduit par les techniques simples issues des Wcsp. Cependant cette réduction a un coût non négligeable lorsque le nombre d'individus ou le nombre de loci est grand.

La seconde partie du chapitre s'est donc intéressée à l'élaboration d'un modèle spécifique aux pedigrees de demi-frères équivalent à la réduction que nous pouvions obtenir.

Le modèle a prouvé son efficacité en terme de qualité de reconstruction et également dans la quantité de données qu'il pouvait traité (36 000 loci en quelques secondes pour des tailles de pedigrees réalistes, 15 descendants).

Lors de la reconstruction des haplotypes du père, nous avons également ajouté une mesure nous permettant d'établir la confiance que nous avions sur la reconstruction de chaque locus.

Conclusion et Perspectives

Nous présentons ici une synthèse des contributions apportées par cette thèse puis nous proposerons plusieurs perspectives de recherche.

Dans cette thèse nous nous sommes placés dans le cadre des réseaux de fonctions de coûts appartenant à la famille des modèles graphiques probabilistes. Notre objectif à travers l'utilisation des techniques d'optimisation et l'élaboration de nouvelles techniques de décomposition était de résoudre le problème de reconstruction d'haplotypes.

Nous avons décrit une décomposition par paire permettant d'exprimer une fonction de coût sous la forme d'une somme de fonctions d'arités plus petites. Son application nous a permis d'une part d'améliorer l'efficacité de la résolution des instances d'analyse de liaison dans le cadre de petits pedigrees issus de la généalogie humaine et d'autre part de réduire, conjointement à l'élimination de variables, le réseau modélisant la reconstruction des haplotypes d'un individu dans un pedigree de demi-frères.

Nous avons pu extraire une modélisation directe formalisant le résultat de la réduction du réseau d'origine. Cette modélisation a obtenu des résultats convaincants en terme de qualité et de quantité de données pouvant être traitées (36 000 loci, 20 individus). Cette modélisation a été intégrée au logiciel `QTLMap`[7]. Au prix de certaines approximations, nous avons montré qu'il est possible de résoudre de manière exacte le problème de grande taille. En vue de son utilisation potentielle dans le cadre de l'inférence nous avons développé une méthode de comptage de solutions dans les réseaux de contraintes qui, associée à une décomposition structurelle des problèmes, permettait de calculer un nombre approché de solutions des problèmes originaux, obtenant des résultats comparables à ceux de l'existant mais bien souvent beaucoup plus rapidement.

Perspectives

La formulation de reconstruction d'haplotypes par le modèle de ségrégation nous amène à avoir des réseaux de grande taille (en nombre de variables, en nombre de fonctions de probabilités). Par exemple, l'INRA possède pour près de 8 000 bovins leur lien de parenté et pour environ 950 d'entre eux leurs génotypes sur 400 loci, ce qui nous amène à avoir un réseau de plus de 11 millions de variables

7. http://www.inra.fr/qtlmap

binaires. La structure complexe du pedigree rend la décomposition fonctionnelle inefficace. Il serait intéressant de rechercher d'autres simplifications du réseau intrinsèque à la nature du problème.

1. Modifier le réseau de ségrégation en diminuant le nombre de variables nécessaires pour chaque individu à chaque locus.
2. Certains individus ne sont pas "utiles" par rapport aux données de génotypes. Par exemple, les individus n'ayant pas dans leur descendance d'individu génotypé ou les fondateurs non génotypés n'ayant qu'un seul enfant (comme dans le cas des mères des pedigrees de demi-frères que nous avons étudié).

Ce dernier point appliqué au pedigree précédent réduirait de plus de la moitié le réseau.

Dans le chapitre 5 nous avons évoqué des perspectives de travail à l'utilisation de la décomposition structurelle pour le comptage des k meilleures solutions d'un réseau de fonctions de coûts (avec l'application pour la reconstruction d'haplotypes au calcul de la distribution de probabilité des configurations d'haplotypes) et pour le calcul de la constante de normalisation dans les réseaux de Markov.

Enfin, nous avons commencé un travail sur la décomposition fonctionnelle approchée, il faudra statuer formellement sur la réciproque du théorème 4.37 et si possible établir un moyen simple de construire les deux fonctions résultantes.

Des expérimentations combinant une décomposition de fonction approchée et une décomposition structurelle approchée pour le problème de reconstruction d'haplotypes sont à faire. L'utilisation d'une décomposition ad-hoc guidée par les familles de demi-frères pourrait être une première piste.

Bibliographie

[Abecasis et al., 2002] Abecasis, G., Cherny, S., Cookson, W., and Cardon, L. (2002). Merlin – rapid analysis of dense genetic maps using sparse gene flow trees. *Nature Genetics*, 30 :97–101.

[Bailleux and Chabrier, 1996] Bailleux, O. and Chabrier, J.-J. (1996). Approximate resolution of hard numbering problems. In *AAAI*.

[Bayardo and Pehoushek, 2000] Bayardo, R. J. and Pehoushek, J. D. (2000). Counting models using connected components. In *Proceedings of 17th National Conference on Artificial Intelligence Research (AAAI'00)*, pages 157–162.

[Bessiere et al., 2009] Bessiere, C., Katsirelos, G., Narodytska, N., and Walsh, T. (2009). Circuit complexity and decompositions of global constraints. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 412–418.

[Birnbaum and Lozinskii, 1999] Birnbaum, E. and Lozinskii, E. L. (1999). The good old Davis-Putnam procedure hepls counting models. *Journal of Artificial Intelligence Research*, 10 :457–477.

[Bistarelli et al., 1999] Bistarelli, S., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G., and Fargier, H. (1999). Semiring-based CSPs and valued CSPs : frameworks, properties, and comparison. *Constraints*, 4.

[Browning and Browning, 2007] Browning, S. and Browning, B. (2007). rapid and accurate haplotype phasing and missing data inference for whole genome association studies using localized haplotype clustering. *American Journal of Human Genetics*, 81(5) :1084–1097.

[Cabon et al., 1999] Cabon, B., de Givry, S., Lobjois, L., Schiex, T., and Warners, J. P. (1999). Radio Link Frequency Assignment. *Constraints*, 4(1) :79–89.

[Cannings et al., 1978] Cannings, C., Thompson, E. A., and Skolnick, M. H. (1978). Probability functions on complex pedigrees. *Advances in Applied Probability*, 10 :26–61.

[Casella and Berger, 2001] Casella, G. and Berger, R. (2001). Statistical inference.

[Cierco-Ayrolles et al., 2004] Cierco-Ayrolles, C., Abdallah, J., Boitard, S., Chikhi, L., Rochambeau, H., and de Tristone, H. (2004). On linkage disequilibrium measures : Methods and applications. *Recent Research Developments in Genetics and Breeding*, 1(1).

[Clark, 1990] Clark, A. G. (1990). Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular biology and Evolution*, 7(2) :111–122.

[Cooper, 1990] Cooper, G. (1990). Computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2) :393–405.

[Cooper, 2003] Cooper, M. C. (2003). Reduction operations in fuzzy or valued constraint satisfaction. *Fuzzy sets ans systems*, 134(3) :311–342.

[Cooper et al., 2008] Cooper, M. C., de Givry, S., Sánchez, M., Schiex, T., and Zytnicki, M. (2008). Virtual arc consistency for weighted csp. In *AAAI*, pages 253–258.

[Cooper et al., 2007] Cooper, M. C., de Givry, S., and Schiex, T. (2007). Optimal soft arc consistency. In *Proceedings of 20th International Joint Conference Artificial Intelligence (IJCAI'07)*, pages 68–73.

[Cooper and Schiex, 2004] Cooper, M. C. and Schiex, T. (2004). Arc consistency for soft constraints. *Artificial Intelligence*, 154(1-2) :199–227.

[Darwiche, 2001] Darwiche, A. (2001). Decomposable negation normal form. *Journal of ACM*, 48 :608–647.

[Darwiche, 2004] Darwiche, A. (2004). New advances in compiling cnf into decomposable negation normal form. In *In Proceedings of European Conference on Artificial Intelligence (ECAI'04)*, pages 328–332.

[Darwiche, 2009] Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, New York, NY, USA, 1st edition.

[Davis and Putnam, 1960] Davis, M. and Putnam, H. (1960). A computing prcedure for quantification theory. *Journal of ACM*, pages 201–215.

[de Givry et al., 2005a] de Givry, S., Bouchez, P., Chabrier, P., Milan, D., and Schiex, T. (2005a). Carthagene : multipopulation integrated genetic and radiated hybrid mapping. *Bioinformatics*, 21(8) :1703–1704.

[de Givry et al., 2005b] de Givry, S., Heras, F., Zytnicki, M., and Schiex, T. (2005b). Existential arc consistency : Getting closer to full arc consistency in weighted CSPs. In *Proceedings of 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 84–89.

[de Givry et al., 2003] de Givry, S., Larrosa, J., Meseguer, P., and Schiex, T. (2003). Solving Max-Sat as weighted CSP. In *Proceeding of the 9th international conference on principles and practice of constraint programming*, LNCS, Kinsale, Ireland. Springer Verlag.

[de Givry et al., 2006] de Givry, S., Schiex, T., and Verfaillie, G. (2006). Exploiting tree decomposition and soft local consistency in weighted csp. In *AAAI*.

[Dearing et al., 1988] Dearing, P., Shier, D., and Warner, D. (1988). Maximal chordal subgraphs. *Discrete Applied Mathematics*, 20(3) :181–190.

[Dechter, 1999] Dechter, R. (1999). Bucket elimination : A unifying framework for reasoning. *Artificial Intelligence*, 113(1–2) :41–85.

[Dechter, 2003] Dechter, R. (2003). *Constraint Programming*. Morgan Kaufmann Publishers.

[Dechter and Mateescu, 2002] Dechter, R. and Mateescu, R. (2002). The impact of AND/OR search spaces on constraint satisfaction and counting. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI'02)*, pages 731–736.

[Dechter and Pearl, 1997] Dechter, R. and Pearl, J. (1997). Structure identification in relational data.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, J. S. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society : Series B*, 39 :1–38.

[Druet and Georges, 2010] Druet, T. and Georges, M. (2010). A Hidden Markov Model Combining Linkage and Linkage Disequilibrium Information for Haplotype Reconstruction and Quantitative Trait Locus Fine Mapping. *Genetics*, 184.

[Elston and Stewart, 1971] Elston, R. C. and Stewart, J. (1971). A general model for the genetic analysis of pedigree data. *Human Heredity*, 21 :523–534.

[Excoffier and Slatkin, 1995] Excoffier, L. and Slatkin, M. (1995). Maximum likelihood estimation of molecular haploptype frequencies in a diploid population. *Molecular Biology and Evolution*, 15(1) :41–60.

[Favier et al., 2011a] Favier, A., de Givry, S., and Jégou, P. (2011a). Solution counting for CSP and SAT with large tree-width. *Control Systems and Computers*, (2) :4–13.

[Favier et al., 2011b] Favier, A., de Givry, S., Legarra, A., and Schiex, T. (2011b). Pairwise decomposition for combinatorial optimization in graphical models. In *IJCAI*, pages 2126–2132.

[Favier et al., 2010] Favier, A., Elsen, J.-M., de Givry, S., and Legarra, A. (2010). Optimal haplotype reconstruction in half-sib families. In *Workshop on Constraint based methods for bioinformatics*.

[Fichelson et al., 2005] Fichelson, M., Dovgolevsky, N., and Geiger, D. (2005). Maximum Likelihood Haplotyping for General Pedigrees. *Human Heredity*, 59(1) :41–60.

[Freuder and Quinn, 1985] Freuder, E. C. and Quinn, M. J. (1985). Taking advantage of stable sets of variables in constraint satisfaction problems. In *In IJCAI'85*, pages 1076–1078.

[Friedman et al., 2000] Friedman, N., Geiger, D., and Lotner, N. (2000). Likelihood computation using value abstraction. In *Proceeding of the* 6^th *Conference on Uncertainty in Artificial Intelligence (UAI'00)*.

[Fulkerson and Gross, 1965] Fulkerson, D. R. and Gross, O. (1965). Incidence matrices and interval graphs. *Pacific Journal Mathematics*, 15 :835–855.

[Gilks et al., 1996] Gilks, W., Richardson, S., and Spiegelhalter, D. (1996). Markov chain monte carlo in practice. *Interdisciplinary Statistics*.

[Gomes et al., 2007] Gomes, C. P., Hoffmann, J., Sabharwal, A., and Selman, B. (2007). From sampling to model counting. In *(IJCAI'07)*, pages 2293–2299.

[Gusfield, 2002] Gusfield, D. (2002). Haplotyping as perfect phylogeny : conceptual framework and efficient solutions. In *International Conference on Research in Computational Malecular Biology*, pages 166–175.

[Haldane, 1919] Haldane, J. (1919). The combination of linkage values and the calculation of distances between the loci of linked factors. *Journal of Genetics*, 8 :299–309.

[Hammersley and Clifford, 1971] Hammersley, J. and Clifford, P. (1971). Markov fields on finite graphs and lattices.

[Haralick and Elliott, 1980] Haralick, R. and Elliott, G. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3) :263–313.

[Hawley and Kidd, 1995] Hawley, M. E. and Kidd, K. K. (1995). Haplo : a programm using the EM algorithm to estimate the frequencies of multi-site haplotypes. *Journal Hereditary*, 85(5) :409–411.

[Heckerman and Breese, 1996] Heckerman, D. and Breese, J. (1996). Causal independence for probability assessment and inference using bayesian networks. *IEEE Systems, Man, and Cyber.*, 26(6) :826–831.

[Jégou and Terrioux, 2003] Jégou, P. and Terrioux, C. (2003). Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence Journal*, 146(1) :43–75.

[Jensen et al., 1990] Jensen, F., Lauritzen, S. L., and Olesen, K. (1990). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quaterly*, 4 :269–282.

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilitic Graphical Models*. The MIT Press.

[Kosambi, 1944] Kosambi, D. (1944). The estimation of map distance from recombination values. Ann. Eugen. 12 :172-175. *Annals of Eugenics*, 12(3) :172–175.

[Kruglyak et al., 1996] Kruglyak, L., Daly, M., Reeve-Daly, M. P., and Lander, E. S. (1996). Parametric and nonparametric linkage analysis : a unified multipoint approach. *American Journal Human Genetics*, 58 :1347–1363.

[Lander and Green, 1987] Lander, E. and Green, P. (1987). Construction of multilocus genetic linkage maps in humans. In *The Proceedings of the National Academy of Sciences U.S.A*, volume 84, pages 2363–2367.

[Lander et al., 1987] Lander, E., Green, P., Abrahamson, J., Barlow, A., Daly, M., Lincoln, S., and Newburg, L. (1987). Mapmaker : an iterative computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics*, 1(2) :174–181.

[Lange and Elston, 1975] Lange, K. and Elston, R. C. (1975). Extensions to pedigree analysis I. Likehood calculations for simple and complex pedigrees. *Human Heredity*, 25(2) :95–105.

[Lange and Goradia, 1987] Lange, K. and Goradia, T. M. (1987). An algorithm for automatic genotype elimination. *American Journal of Human Genetics*, 40 :250–256.

[Larrosa, 2000] Larrosa, J. (2000). Boosting search with variable elimination. In *Proceeding of 6th International Conference of Principles and Practice of Constraint Programming (CP'00)*, pages 291–305.

[Larrosa, 2002] Larrosa, J. (2002). Node and arc consistency in weighted CSP. *Proceedings of the National Conference on Artificial Intelligence (AAAI'02).*

[Larrosa and Schiex, 2003] Larrosa, J. and Schiex, T. (2003). In the quest of the best form of local consistency for Weighted CSP. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'03).*

[Lauritzen, 1996] Lauritzen, S. L. (1996). *Graphical Models.* Oxford : Clarendon Press.

[Lauritzen and Sheehan, 2003] Lauritzen, S. L. and Sheehan, N. A. (2003). Graphical models for genetics analyses. *Statistical Science*, 18(4) :489–514.

[Lauritzen and Spiegelhalter, 1988] Lauritzen, S. L. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society : Series B*, 50(2) :157–224.

[Lecoutre et al., 2006] Lecoutre, C., Sais, L., Tabary, S., and Vidal, V. (2006). Last conflict based reasoning. In *ECAI*, pages 133–137.

[Lecoutre et al., 2009] Lecoutre, C., Saïs, L., Tabary, S., and Vidal, V. (2009). Reasoning from last conflict(s) in constraint programming. 173 :1592,1614.

[Lee and Leung, 2009] Lee, J. and Leung, K. L. (2009). Towards efficient consistency enforcement for global constraints in weighted constraint satisfaction. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 559–565.

[Lerman and Rouat, 1999] Lerman, I. C. and Rouat, V. (1999). Segmentation de la sériation pour la résolution de #SAT. *Mathématiques et Sciences Humaines*, 147 :113–134.

[Li and Jiang, 2003] Li, J. and Jiang, T. (2003). efficient inference of haplotypes from genotypes on a pedigree. *Journal of Bioinformatics and Computational Biology*, 1(1) :41–69.

[Mackworth, 1977] Mackworth, A. K. (1977). On reading sketch maps. In *IJCAI*, pages 598–606.

[Marinescu and Dechter, 2009] Marinescu, R. and Dechter, R. (2009). Memory intensive and/or search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17) :1492–1524.

[Meiri et al., 1990] Meiri, I., Dechter, R., and Pearl, J. (1990). Tree decomposition with applications to constraint processing. In *Proceeding of AAAI'90*, pages 10–16.

[Mendel, 1866] Mendel, G. (1866). Experiment in Plant Hybridisation.

[Montanari, 1974] Montanari, U. (1974). Networks of constraints : fundamental properties and application to picture processing. *Information Science*, 7 :95–132.

[Niu, 2004] Niu, T. (2004). Algorithms for Inferring Haplotypes. *Genetic EPidemiology*, 27 :334–347.

[O'Connell, 2000] O'Connell, J. R. (2000). Zero-REcombinant haplotyping : applications to fine mapping using SNPs. *Genetic Epidemiology*, 19 :64–74.

[Pearl, 1988] Pearl, J. (1988). *Probablistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

[Pesant, 2005] Pesant, G. (2005). Counting solutions of CSPs : a structural approach. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 260–265.

[Qian and Beckmann, 2002] Qian, D. and Beckmann, L. (2002). Minimum recombinant haplotyping in pedigrees. *American Journal of Human Genetics*, 70(6) :1434–1445.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected application in speech recognition. In *IEEE*, volume 77, pages 257–286.

[Rollon et al., 2011] Rollon, E., Flevora, N., and Dechter, R. (2011). Inference schemes for m best solutions for soft csps. *Proceedings of workshop Soft 2011*.

[Sabin and Freuder, 1994] Sabin, D. and Freuder, E. (1994). Contradicting Conventional Wisdom in Constraint Satisfaction. In *Proc. of ecai*, pages 125–129.

[Sanchez et al., 2008] Sanchez, M., de Givry, S., and Schiex, T. (2008). Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. *Constraints*, 13(1) :130–154.

[Sánchez et al., 2004] Sánchez, M., Meseguer, P., and Larrosa, J. (2004). Using constraints with memory to implement variable elimination. In *ECAI*, pages 216–220, Spain.

[Sang et al., 2004] Sang, T., Bacchus, F., Beame, P., Kautz, H. A., and Pitassi, T. (2004). Combining component caching and clause learning for effective model counting. *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*.

[Savicky and Vomlel, 2007] Savicky, P. and Vomlel, J. (2007). Exploiting tensor rank-one decomposition in probabilistic inference. *Kybernetika*, 43(5) :747–764.

[Schiex, 2000] Schiex, T. (2000). Arc consistency for Soft Constraints. In *Proceedings of Principles and Practice of Constraint Programming (CP'00)*.

[Sobel and Lange, 1993] Sobel, E. and Lange, K. (1993). Metropolis sampling in pedigree analysis. *Statistitical Methods in Medical Research*, 2(3) :263–282.

[Sobel et al., 1996] Sobel, E., Lange, K., O'Connell, J., and Weeks, D. (1996). Haplotyping algorithms. In Speed, T. and Waterman, M. S., editors, *Genetic mapping and DNA sequencing*, volume 81, pages 89–110. Springer-Verlag.

[Stam, 1993] Stam, P. (1993). Construction of intergrated genetic linkage maps by means of a new computer package : Joinmap. *The Plant Journal*, 3 :739–744.

[Stephens and Donelly, 2000] Stephens, M. and Donelly, P. (2000). Inference in molecular genetics. *Journal of the Royal Statistical Society : Series B*, 62 :605–655.

[Tarjan and Yannakakis, 1984] Tarjan, R. and Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3) :566–579.

[Thomas, 1985] Thomas, A. (1985). *Data structures, methods of approximation and optimal computation for pedigree analysis.* PhD thesis, Cambridge University.

[Thompson, 1994] Thompson, E. A. (1994). Monte carlo likelihood in genetic mapping. *Statistical Science*, 9(3) :355–366.

[Thurley, 2006] Thurley, M. (2006). sharpSAT - Counting Models with Advanced Component Caching and Implicit BCPs. *Proceedings of the 9th International Conference on Theory and Applications of Satifiability Testing (SAT'06)*, pages 424–429.

[Valiant, 1979] Valiant, L. (1979). The complexity of computing the permanent. *Theoretical Computer Sciences*, 8 :189–201.

[Wei and Selman, 2005] Wei, W. and Selman, B. (2005). A new approach to model counting. In *Proceeding of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, pages 324–339.

[Wexler and Meek, 2008] Wexler, Y. and Meek, C. (2008). Mas : a multiplicative approximation scheme for probabilistic inference. In *Proceedings of the 22th Annual Conference on Neural Information Processing Systems (NIPS'08)*, pages 1761–1768.

[Wijsman, 1987] Wijsman, E. (1987). A deductive method of haplotype analysis in pedigrees. *American Journal Human Genetics*, 41 :356–373.

[Windig and Meuwissen, 2004] Windig, J. and Meuwissen, T. (2004). Rapid haplotype reconstruction in pedigrees with dense marker maps. *J. of Animal Breeding and Genetics*, 121 :26–39.

[Zhang et al., 2005] Zhang, K., Sun, F., and Zhao, H. (2005). HAPLORE : a program for haplotype reconstruction in general pedigrees without reconbination. *Bioinformatics*, 21 :90–103.

[Zhang and Poole, 1996] Zhang, N. L. and Poole, D. (1996). Exploiting Causal Independence in Bayesian Network Inference. *Journal of Artificial Intelligence Research*, pages 301–328.

[Zhang et al., 2008] Zhang, Y., Yap, R. H. C., Li, C., and Marisetti, S. (2008). An elimination algorithm for functional constraints. In *Proceedings of Principles and Practice of Constraint Programming (CP'08)*, pages 545–549.

Réduction du modèle de ségrégation : détails

Dans cette annexe, il s'agit de détailler les calculs réalisés à chaque étape pour l'expression de F_i suivante :

$$\begin{aligned} F_i = \sum_{\texttt{Fam}_i} \Big(\prod_{k=1}^{|L|} & \mathbb{P}(A^P_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{\texttt{mere}(i)k}) \\ & \times \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik}) \\ & \times \mathbb{P}(G^i_k \mid A^P_{ik}, A^M_{ik}) \cdot \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \cdot \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)}) \Big) \end{aligned} \tag{A.1}$$

Etape 1 : Elimination des variables associées à la mère m

Les sommes sur les variables associées à la mère ($\mathcal{M}_i$) sont indépendantes des produits faits sur les loci F_i peut se formuler de la manière suivante :

$$\begin{aligned} F_i = \sum_{\mathcal{F}_i} \Big(\prod_{k=1}^{|L|} & [\mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot \mathbb{P}(G^i_k \mid A^P_{ik}, A^M_{ik}) \cdot \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \cdot \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)})] \\ & \times \prod_{k=1}^{|L|} \sum_{\mathcal{M}_i} [\mathbb{P}(A^P_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik})] \Big) \end{aligned}$$

Soit $C^i_1 = \sum_{\mathcal{M}_i} \prod_{k=1}^{|L|} [\mathbb{P}(A^P_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik})]$. Les fréquences alléliques sont supposées équiprobables donc $\mathbb{P}(A^P_{\texttt{mere}(i)k}) = \mathbb{P}(A^M_{\texttt{mere}(i)k}) = \frac{1}{2}$ à tous les loci. D'où $C^i_1 = \left(\frac{1}{4}\right)^{|L|} \sum_{\mathcal{M}_i} \prod_{k=1}^{|L|} [\mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik})]$. La table A.1 page suivante permet de déduire que $C^i_1 = \left(\frac{1}{2}\right)^{|L|}$.

S^M_{ik}	A^M_{ik}	$A^P_{\texttt{mere}(i)}$	$A^M_{\texttt{mere}(i)}$	$\mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik})$	$\sum_{\substack{A^P_{\texttt{mere}(i)k} \\ A^M_{\texttt{mere}(i)k}}} \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik})$
P	a	a	a	1	2
			b	1	
		b	a	0	
			b	0	
	b	a	a	0	2
			b	0	
		b	a	1	
			b	1	
M	a	a	a	1	2
			b	0	
		b	a	1	
			b	0	
	b	a	a	0	2
			b	1	
		b	a	0	
			b	1	

Table A.1 – Détail du calcul pour un locus k de $\sum_{A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}} \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik})$

L'expression de F_i devient

$$F_i = C_1^i \cdot \sum_{\mathcal{F}_i} \prod_{k=1}^{|L|} \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot \mathbb{P}(G^i_k \mid A^P_{ik}, A^M_{ik}) \cdot \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \cdot \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)})$$

Etape 2 : Elimination des variables 'maternelles' associées au descendant i

Les sommes sur les variables des ensembles $\mathcal{S}_i^M$ et $\mathcal{A}_i^M$ sont indépendantes des produits faits sur les loci, ainsi F_i peut se formuler de la manière suivante :

$$\begin{aligned} F_i = \quad & C_1^i \cdot \sum_{\mathcal{F}_i \setminus \mathcal{S}_i^M \cup \mathcal{A}_i^M} \prod_{k=1}^{|L|} \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \\ & \times \sum_{\mathcal{A}_i^M} \prod_{k=1}^{|L|} \mathbb{P}(G^i_k \mid A^P_{ik}, A^M_{ik}) \\ & \times \sum_{\mathcal{S}_i^M} \prod_{k=1}^{|L|} \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)}) \end{aligned}$$

Par la loi fondamentale généralisée des probabilités (définition 2.12 page 20) et le fait qu'il n'y a aucune observation sur ces variables nous avons $\prod_{k=1}^{|L|} \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)}) = \mathbb{P}(S^M_{i1}, S^M_{i2}, \ldots, S^M_{i|L|})$ par conséquent

$\sum_{\mathcal{S}_i^M} \prod_{k=1}^{|L|} \mathbb{P}(S_{ik}^M \mid S_{i(k-1)}^M) = 1.$

L'expression $\sum_{\mathcal{A}_i^M} \prod_{k=1}^{|L|} \mathbb{P}(G_k^i \mid A_{ik}^P, A_{ik}^M)$ devient $\prod_{k=1}^{|L|} \sum_{A_{ik}^M} \mathbb{P}(G_k^i \mid A_{ik}^P, A_{ik}^M) = \prod_{k=1}^{|L|} h(G_k^i, A_{ik}^P)$

G_k^i étant une observation, h est fonction de A_{ik}^P et paramétrée par G_k^i : $h_{G_k^i}(A_{ik}^P)$. La table suivante donne la définition de cette fonction h.

G_k^i \ A_{ik}^P	a	b
aa	1	0
ab	1	1
bb	0	1

Table A.2 – $h_{G_k^i}(A_{ik}^P)$

$$F_i = C_1^i \cdot \sum_{\mathcal{F}_i \setminus \mathcal{S}_i^M \cup \mathcal{A}_i^M} \prod_{k=1}^{|L|} \mathbb{P}(A_{ik}^P \mid A_{\boldsymbol{p}k}^P, A_{\boldsymbol{p}k}^M, S_{ik}^P) \cdot \mathbb{P}(S_{ik}^P \mid S_{i(k-1)}^P) \cdot h_{G_k^i}(A_{ik}^P)$$

Etape 3 : Fusion des variables allèles du père p

La fusion de certains des couples variables $(A_{\boldsymbol{p}1}^P, A_{\boldsymbol{p}1}^M), \ldots, (A_{\boldsymbol{p}|L|}^P, A_{\boldsymbol{p}|L|}^M)$, est intéressant car se sont des variables booléennes, et leur lien peut être assimilé à des contraintes de différences ou d'égalités suivant les cas. En effet si le génotype observé au locus k est aa alors $A_{\boldsymbol{p}1}^P = A_{\boldsymbol{p}1}^M$ et si le génotype est ab (ou ba) alors $A_{\boldsymbol{p}1}^P \neq A_{\boldsymbol{p}1}^M$.

Soit H_k la variable haplotype résultant de la fusion de $A_{\boldsymbol{p}k}^P$ et $A_{\boldsymbol{p}k}^M$ pour chaque loci k étudié.

Nous allons voir que certaines variables H_k ne sont pas nécessaires et comment sont réellement définies les autres.

Pour chaque locus k nous avons $\mathbb{P}(A_{\boldsymbol{p}k}^P) \cdot \mathbb{P}(A_{\boldsymbol{p}k}^M) \cdot \mathbb{P}(G_k^{\boldsymbol{p}} \mid A_{\boldsymbol{p}k}^P, A_{\boldsymbol{p}k}^M)$

1er cas : le père est homozygote, $G_k^{\boldsymbol{p}} = aa$ (resp. $G_k^{\boldsymbol{p}} = bb$) au locus k

Les variables $A_{\boldsymbol{p}k}^P$ et $A_{\boldsymbol{p}k}^M$) peuvent être affectées sans ambiguïté $A_{\boldsymbol{p}k}^P = A_{\boldsymbol{p}k}^M) = a$ (resp. b). Il n'est pas nécessaire dans ce cas là de créer une nouvelle variable.

2e cas : le père est hétérozygote ab au locus k

La variable H_k a comme domaine $\left\{\frac{a}{b}, \frac{b}{a}\right\}$ tel que $H_k = \begin{cases} \frac{a}{b} & \text{si } A_{\boldsymbol{p}k}^P = a, A_{\boldsymbol{p}k}^M = b \\ \frac{b}{a} & \text{si } A_{\boldsymbol{p}k}^P = b, A_{\boldsymbol{p}k}^M = a \end{cases}$ et $p(H_k = \frac{a}{b}) = P(A_{\boldsymbol{p}k}^P = a) \cdot \mathbb{P}(A_{\boldsymbol{p}k}^M = b) \cdot \mathbb{P}(G_k^{\boldsymbol{p}} = ab \mid A_{\boldsymbol{p}k}^P = a, A_{\boldsymbol{p}k}^M = b) = 0.25$

$p(H_k = \frac{b}{a}) = P(A_{\boldsymbol{p}k}^P = b) \cdot \mathbb{P}(A_{\boldsymbol{p}k}^M = a) \cdot \mathbb{P}(G_k^{\boldsymbol{p}} = ab \mid A_{\boldsymbol{p}k}^P = b, A_{\boldsymbol{p}k}^M = a) = 0.25$

En résumé, pour les loci hétérozygotes nous avons remplacé les variables $A^P_{\boldsymbol{p}k}$ et $A^M_{\boldsymbol{p}k}$ par H_k, pour les autres (loci homozygotes) nous les avons conservées.

Ainsi nous avons

$$\sum_{\mathbf{V}\setminus\mathcal{P}} \mathbb{P}(\mathbf{V}) = (0.25)^{|L|-|\mathcal{H}_{\boldsymbol{p}}|} \prod_{k\in\mathcal{H}_{\boldsymbol{p}}} \mathbb{P}(H_k) \times \prod_{i=1}^{n} \sum_{\texttt{Fam}_i} F_i$$

avec

$$\begin{aligned} F_i = \quad & C^i_1 \cdot \sum_{\mathcal{F}_i\setminus\mathcal{S}^M_i\cup\mathcal{A}^M_i} \prod_{k=1}^{|L|} \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \\ & \times \prod_{k\in\mathcal{H}_{\boldsymbol{p}}} \mathbb{P}(A^P_{ik} \mid H_k, S^P_{ik}) \cdot h_{G^i_k}(A^P_{ik}) \\ & \times \prod_{k\notin\mathcal{H}_{\boldsymbol{p}}} \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot h_{G^i_k}(A^P_{ik}) \end{aligned}$$

Etape 4 : Elimination des variables allèles paternels associées au descendant i.

A ce stade il reste plus qu'à sommer les dernières variables associées au descendant i. $\mathcal{F}_i \setminus \mathcal{S}^M_i \cup \mathcal{A}^M_i = \mathcal{S}^P_i \cup \mathcal{A}^P_i$

Dans un premier temps nous réorganisons l'expression de F_i pour mettre en évidence les différentes phases de sommation à réaliser.

$$\begin{aligned} F_i = C^i_1 \cdot \quad & \sum_{\mathcal{S}^P_i} \prod_{k=1}^{|L|} \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \\ & \times \prod_{k\in\mathcal{H}_{\boldsymbol{p}}} \sum_{A^P_{ik}} \mathbb{P}(A^P_{ik} \mid H_k, S^P_{ik}) \cdot h_{G^i_k}(A^P_{ik}) \\ & \times \prod_{k\notin\mathcal{H}_{\boldsymbol{p}}} \sum_{A^P_{ik}} \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot h_{G^i_k}(A^P_{ik}) \end{aligned}$$

Lorsque le père est homozygote il est simple de calculer

$$\prod_{k\notin\mathcal{H}_{\boldsymbol{p}}} \sum_{A^P_{ik}} \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot h_{G^i_k}(A^P_{ik})$$

En effet par la nature de la table de probabilités de $\mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik})$ (*cf.* table 7.7a page 137) nous avons

$$\prod_{k\notin\mathcal{H}_{\boldsymbol{p}}} \sum_{A^P_{ik}} \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot h_{G^i_k}(A^P_{ik}) = \prod_{k\notin\mathcal{H}_{\boldsymbol{p}}} \sum_{A^P_{ik}} \delta_{A^P_{ik}A^P_{\boldsymbol{p}k}} \cdot h_{G^i_k}(A^P_{ik})$$

où $\delta_{A^P_{ik}A^P_{\boldsymbol{p}k}}$ est le symbole de Kronecker. Puisque le père est homozygote au locus étudié la variable allèle A^P_{ik} est indépendante de la variable de ségrégation S^P_{ik}.

Finalement, $\prod_{k\notin\mathcal{H}_{\boldsymbol{p}}} \sum_{A^P_{ik}} \delta_{A^P_{ik}A^P_{\boldsymbol{p}k}} \cdot h_{G^i_k}(A^P_{ik}) = C^i_2$ où C^i_2 est une constante par rapport aux variables de $\mathcal{P}$.

Lorsque le père $\boldsymbol{p}$ est hétérozygote nous pouvons encore distinguer deux cas : le fils est homozygote ou le fils est hétérozygote au locus :

$$\begin{aligned}\prod_{k\in\mathcal{H}_{\boldsymbol{p}}}\sum_{A_{ik}^P}\mathbb{P}(A_{ik}^P \mid H_k, S_{ik}^P)\cdot h_{G_k^i}(A_{ik}^P) \;&=\; \prod_{k\in\mathcal{H}_{\boldsymbol{p}}\cap\mathcal{H}_i}\sum_{A_{ik}^P}\mathbb{P}(A_{ik}^P \mid H_k, S_{ik}^P)\cdot h_{G_k^i}(A_{ik}^P)\\ &\times \prod_{k\in\mathcal{H}_{\boldsymbol{p}}\setminus\mathcal{H}_i}\sum_{A_{ik}^P}\mathbb{P}(A_{ik}^P \mid H_k, S_{ik}^P)\cdot h_{G_k^i}(A_{ik}^P)\end{aligned}$$

Lorsque le père et le fils sont hétérozygotes au locus k, $h_{G_k^i}(A_{ik}^P)=1$ pour chaque valeur de A_{ik}^P,

$$\prod_{k\in\mathcal{H}_{\boldsymbol{p}}\cap\mathcal{H}_i}\sum_{A_{ik}^P}\mathbb{P}(A_{ik}^P \mid H_k, S_{ik}^P)\cdot h_{G_k^i}(A_{ik}^P) \;=\; \prod_{k\in\mathcal{H}_{\boldsymbol{p}}\cap\mathcal{H}_i}\sum_{A_{ik}^P}\mathbb{P}(A_{ik}^P \mid H_k, S_{ik}^P) \;=\; 1$$

Finalement,

$$\begin{aligned}\prod_{k\in\mathcal{H}_{\boldsymbol{p}}}\sum_{A_{ik}^P}\mathbb{P}(A_{ik}^P \mid H_k, S_{ik}^P)\cdot h_{G_k^i}(A_{ik}^P) \;&=\; \prod_{k\in\mathcal{H}_{\boldsymbol{p}}\setminus\mathcal{H}_i}\sum_{A_{ik}^P}\mathbb{P}(A_{ik}^P \mid H_k, S_{ik}^P)\cdot h_{G_k^i}(A_{ik}^P)\\ &=\; \prod_{k\in\mathcal{H}_{\boldsymbol{p}}\setminus\mathcal{H}_i} g_{G_k^i}(H_k, S_{ik}^P)\end{aligned}$$

$$F_i = C_1^i\cdot C_2^i\sum_{\mathcal{S}_i^P}\prod_{k=1}^{|L|}\mathbb{P}(S_{ik}^P \mid S_{i(k-1)}^P) \quad\cdot \prod_{k\in\mathcal{H}_{\boldsymbol{p}}\setminus\mathcal{H}_i} g_{G_k^i}(H_k, S_{ik}^P) \tag{A.2}$$

S_{ik}^P \ H_k	$\frac{a}{b}$	$\frac{b}{a}$
P	1	0
M	0	1

(a) $G_k^i = aa$

S_{ik}^P \ H_k	$\frac{a}{b}$	$\frac{b}{a}$
P	0	1
M	1	0

(b) $G_k^i = bb$

Table A.3 – $g_{G_k^i}(H_k, S_{ik}^P)$

Etape 5 : Elimination des variables de ségrégation paternelle associées au descendant i

$$F_i = C_1^i\cdot C_2^i\sum_{\mathcal{S}_i^P}\prod_{k\in\mathcal{H}_{\boldsymbol{p}}\setminus\mathcal{H}_i} g_{G_k^i}(H_k, S_{ik}^P) \quad\times\prod_{k=1}^{|L|}\mathbb{P}(S_{ik}^P \mid S_{i(k-1)}^P) \tag{A.3}$$

$$F_i = C_1^i\cdot C_2^i\sum_{\substack{S_{ij}^P\\ j\in\mathcal{H}_{\boldsymbol{p}}\setminus\mathcal{H}_i}}\prod_{k\in\mathcal{H}_{\boldsymbol{p}}\setminus\mathcal{H}_i} g_{G_k^i}(H_k, S_{ik}^P) \quad\times\sum_{\substack{S_{ij}^P\\ j\notin\mathcal{H}_{\boldsymbol{p}}\setminus\mathcal{H}_i}}\prod_{k=1}^{|L|}\mathbb{P}(S_{ik}^P \mid S_{i(k-1)}^P)$$

1) $\underline{S^P_{ik} \textbf{ avec } k \notin \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i}$

Puisque $k \notin \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$ donc $k \in L \setminus (\mathcal{H}_p \setminus \mathcal{H}_i)$.
Soit $L \setminus (\mathcal{H}_p \setminus \mathcal{H}_i) = \{k_1, k_2, \ldots, k_{|\mathcal{L} \setminus \mathcal{H}_i|}\}$ tel que $\forall x < y$, $k_x < k_x$.
Nous avons $\sum_{S^P_{ik_1}} \mathbb{P}(S^P_{i(k_1+1)} \mid S^P_{ik_1}) \cdot \mathbb{P}(S^P_{i(k_1)} \mid S^P_{i(k_1-1)}) = \mathbb{P}(S^P_{i(k_1+1)} \mid S^P_{i(k_1-1)})$ De manière itérative on somme sur $S^P_{i(k_2)}$, $S^P_{i(k_3)}$, etc.
Ce qui permet d'obtenir,

$$\sum_{\substack{S^P_{ik} \\ k \notin \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i}} \prod_{k=1}^{|L|} \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) = \prod_{k \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i} \mathbb{P}(S^P_{ik} \mid S^P_{il_k})$$

avec $l_k < k, l_k \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$ et $\nexists m \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i$ tels que $l_k < m < k$

2) $\underline{S^P_{ik} \textbf{ avec } k \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i}$

Soit $\mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i = \{m_1, m_2, \ldots, m_M\}$ tel que $\forall x < y$, $m_x < m_y$ et $M = \mid \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i \mid$.

$$\sum_{\substack{S^P_{ij} \\ k \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i}} \prod_{k=1}^{M} g_{G^i_k}(H_{m_k}, S^P_{im_k}) \mathbb{P}(S^P_{im_k} \mid S^P_{im_{k-1}})$$

$$= \sum_{S^P_{im_M}} g_{G^i_{m_M}}(H_{m_M}, S^P_{im_M})$$

$$\sum_{S^P_{im_{M-1}}} g_{G^i_{m_{M-1}}}(H_{m_{M-1}}, S^P_{im_{M-1}}) \cdot \mathbb{P}(S^P_{i,m_M} \mid S^P_{i,m_{M-1}})$$

$$\sum_{S^P_{im_{M-2}}} g_{G^i_{m_{M-2}}}(H_{m_{M-2}}, S^P_{im_{M-2}}) \cdot \mathbb{P}(S^P_{i,m_{M-1}} \mid S^P_{i,m_{M-2}})$$

$$\sum_{S^P_{im_{M-3}}} \cdots \sum_{S^P_{im_2}} g_{G^i_{m_2}}(H_{m_2}, S^P_{im_2}) \cdot \mathbb{P}(S^P_{im_3} \mid S^P_{i,m_2})$$

$$\sum_{S^P_{im_1}} g_{G^i_{m_1}}(H_{m_1}, S^P_{im_1}) \cdot \mathbb{P}(S^P_{im_2} \mid S^P_{i,m_1}) \cdot \mathbb{P}(S^P_{i,m_1})$$

Propriété A.1 *Soit la suite* $(u_k)_{k \in [0,M]}$ $u_k = \sum_{S^P_{i,m_k}} \mathbb{P}(S^P_{i,m_{k+1}} \mid S^P_{i,m_k}) \cdot g_{G^i_{m_k}}(H_{m_k}, S^P_{im_k}) \cdot u_{k-1}$ *et* $u_0 = \mathbb{P}(S^P_{im_1})$. *Il existe deux fonctions* f *et* t *telles que*

1. $u_1 = \frac{1}{2} \cdot t_{G^i_{m_1}}(H_{m_1}, S^P_{i,m_2})$
2. $u_k = t_{G^i_{m_k}}(H_{m_k}, S^P_{i,m_{k+1}}) \times \prod_{j=2}^{k} f(H_{m_{j-1}}, H_{m_j}), \quad \forall k \in [2, M[$
3. $u_M = \prod_{k=2}^{M} f(H_{m_{k-1}}, H_{m_k})$

Démonstration.

$$\begin{aligned} u_1 = & \sum_{S^P_{im_1}} \mathbb{P}(S^P_{im_1}) \cdot \mathbb{P}(S^P_{im_2} \mid S^P_{i,m_1}) \cdot g_{G^i_{m_k}}(H_{m_1}, S^P_{im_1}) \\ = & \frac{1}{2} \sum_{S^P_{im_1}} \mathbb{P}(S^P_{im_2} \mid S^P_{im_1}) \cdot g_{G^i_{m_1}}(H_{m_1}, S^P_{im_1}) \\ = & \frac{1}{2} t_{G^i_{m_1}}(H_{m_1}, S^P_{im_2}) \end{aligned}$$

Je détaille la construction de $t_{G^i_{m_1}}(H_{m_1}, S^P_{im_2})$ le cas $G^i_{m_1} = aa$ à travers la table A.4 suivante :

C'est exactement le même raisonnement pour le cas $G^i_{m_1} = bb$, et la fonction $t_{G^i_k}(H_k, S^P_{k+1})$ est définie

H_{m_1}	$S^P_{im_2}$	$S^P_{im_1}$	$\mathbb{P}(S^P_{im_2} \mid S^P_{im_1}) \cdot g_{aa}(H_{m_1}, S^P_{im_1})$	$t_{aa}(H_{m_1}, S^P_{im_2})$
$\frac{a}{b}$	P	P M	$1 - r_{m_1m_2}$ 0	$1 - r_{m_1m_2}$
	M	P M	0 $r_{m_1m_2}$	$r_{m_1m_2}$
$\frac{b}{a}$	P	P M	0 $r_{m_1m_2}$	$r_{m_1m_2}$
	M	P M	$1 - r_{m_1m_2}$ 0	$1 - r_{m_1m_2}$

Table A.4

par les tables A.5a et A.5b.

S^P_{k+1} \ H_k	$\frac{a}{b}$	$\frac{b}{a}$
P	$1 - r_{m_1m_2}$	$r_{m_1m_2}$
M	$r_{m_1m_2}$	$1 - r_{m_1m_2}$

(a) G^i_k=aa

S^P_{k+1} \ H_k	$\frac{a}{b}$	$\frac{b}{a}$
P	$r_{m_1m_2}$	$1 - r_{m_1m_2}$
M	$1 - r_{m_1m_2}$	$r_{m_1m_2}$

(b) G^i_k=bb

Table A.5 – $t_{G^i_k}(H_k, S^P_{k+1})$

Soit le locus m_k, supposons que $\forall\, 1 < l < k$, $u_l = f_{G^i_{m_{l-1}}, G^i_{m_l}}(H_{m_{l-1}}, H_{m_l}) \times t_{G^i_{m_l}}(H_{m_l}, S^P_{i,m_{l+1}})$

$$u_k = \sum_{S^P_{i,m_k}} \mathbb{P}(S^P_{i,m_{k+1}} \mid S^P_{i,m_k}) \cdot g_{G^i_{m_k}}(H_{m_k}, S^P_{im_k}) \cdot u_{k-1}$$

$$u_k = \sum_{S^P_{i,m_k}} \mathbb{P}(S^P_{i,m_{k+1}} \mid S^P_{i,m_k}) \cdot g_{G^i_{m_k}}(H_{m_k}, S^P_{im_k}) \cdot \prod_{l=2}^{k} f(H_{m_{l-2}}, H_{m_{l-1}}) \times t_{G^i_{m_{k-1}}}(H_{m_{k-1}}, S^P_{i,m_k})$$

$$u_k = \prod_{l=2}^{k-1} f(H_{m_{l-1}}, H_{m_l}) \cdot \sum_{S^P_{i,m_k}} \mathbb{P}(S^P_{i,m_{k+1}} \mid S^P_{i,m_k}) \cdot g_{G^i_{m_k}}(H_{m_k}, S^P_{im_k}) \cdot t_{G^i_{m_{k-1}}}(H_{m_{k-1}}, S^P_{i,m_k})$$

L'application de la décomposition par paire sur la table A.6 permet de déduire

$$u_k = \prod_{l=2}^{k} f_{G^i_{m_{l-1}}, G^i_{m_l}}(H_{m_{l-1}}, H_{m_l}) \times t_{G^i_{m_k}}(H_{m_k}, S^P_{i,m_{k+1}})$$

$H_{m_{i-1}}$	H_{m_i}	$S^P_{im_{k+1}}$	$S^P_{im_k}$	$\mathbb{P}(S^P_{i,m_{k+1}} \mid S^P_{i,m_k})$ $\times g_{G^i_{m_k}}(H_{m_k}, S^P_{im_k})$ $\times t_{G^i_{m_{k-1}}}(H_{m_{k-1}}, S^P_{i,m_k})$	u_k
$\frac{a}{b}$	$\frac{a}{b}$	P	P	$(1 - r_{m_k m_{k+1}}) \cdot (1 - r_{m_{k-1} m_k})$	$(1 - r_{m_{k-1} m_k}) \cdot (1 - r_{m_{k-1} m_k})$
			M	0	
		M	P	$r_{m_k m_{k+1}} \cdot (1 - r_{m_{k-1} m_k})$	$r_{m_{k-1} m_k} \cdot (1 - r_{m_{k-1} m_k})$
			M	0	
	$\frac{b}{a}$	P	P	0	$r_{m_{k-1} m_k} \cdot r_{m_{k-1} m_k}$
			M	$r_{m_{k-1} m_k} \cdot r_{m_{k-1} m_k}$	
		M	P	0	$(1 - r_{m_{k-1} m_k}) \cdot r_{m_{k-1} m_k}$
			M	$(1 - r_{m_{k-1} m_k}) \cdot r_{m_{k-1} m_k}$	
$\frac{b}{a}$	$\frac{a}{b}$	P	P	$(1 - r_{m_k m_{k+1}}) \cdot r_{m_{k-1} m_k}$	$(1 - r_{m_{k-1} m_k}) \cdot r_{m_{k-1} m_k}$
			M	0	
		M	P	$r_{m_k m_{k+1}} \cdot r_{m_{k-1} m_k}$	$r_{m_{k-1} m_k} \cdot r_{m_{k-1} m_k}$
			M	0	
	$\frac{b}{a}$	P	P	0	$r_{m_{k-1} m_k} \cdot (1 - r_{m_{k-1} m_k})$
			M	$r_{m_{k-1} m_k} \cdot (1 - r_{m_{k-1} m_k})$	
		M	P	0	$(1 - r_{m_{k-1} m_k}) \cdot (1 - r_{m_{k-1} m_k})$
			M	$(1 - r_{m_{k-1} m_k}) \cdot (1 - r_{m_{k-1} m_k})$	

Table A.6 – Détail du calcul de u_k pour $O_{im_k} = aa$ et $G^i_{m_{k-1}} = aa$

et

$$f_{G^i_{m_{k-1}}, G^i_{m_k}}(H_{m_{k-1}}, H_{m_k}) = \begin{cases} (1 - r_{m_{k-1} m_k}) & \text{si } (G^i_{m_{k-1}} = G^i_{m_k}) \text{ et } (H_{m_{k-1}} = H_{m_k}) \\ & \text{ou } (G^i_{m_{k-1}} \neq G^i_{m_k}) \text{ et } (H_{m_{k-1}} \neq H_{m_k}) \\ r_{m_{k-1} m_k} & \text{si } (G^i_{m_{k-1}} = G^i_{m_k}) \text{ et } (H_{m_{k-1}} \neq H_{m_k}) \\ & \text{ou } (G^i_{m_{k-1}} \neq G^i_{m_k}) \text{ et } (H_{m_{k-1}} = H_{m_k}) \end{cases}$$

$H_{m_{M-1}}$	H_{m_M}	$S^P_{im_M}$)	$g_{G^i_{m_M}}(H_{m_M}, S^P_{im_M})$ $\times t_{G^i_{m_{M-1}}}(H_{m_{M-1}}, S^P_{i,m_M})$	$\sum_{S^P_{i,m_M}} g_{G^i_{m_M}}(H_{m_M}, S^P_{im_M})$ $\times t_{G^i_{m_{M-1}}}(H_{m_{M-1}}, S^P_{i,m_M})$
$\frac{a}{b}$	$\frac{a}{b}$	P M	0 $r_{m_{M-1}m_M}$	$r_{m_{M-1}m_M}$
	$\frac{b}{a}$	P M	0 $(1 - r_{m_{M-1}m_M})$	$(1 - r_{m_{M-1}m_M})$
$\frac{b}{a}$	$\frac{a}{b}$	P M	$(1 - r_{m_{M-1}m_M})$ 0	$(1 - r_{m_{M-1}m_M})$
	$\frac{b}{a}$	P M	$r_{m_{M-1}m_M}$ 0	$r_{m_{M-1}m_M}$

Table A.7 – Détail du calcul de $\sum_{S^P_{i,m_M}} g_{G^i_{m_M}}(H_{m_M}, S^P_{im_M}) \cdot t_{G^i_{m_M}}(H_{m_{M-1}}, S^P_{i,m_M})$ avec $G^i_{m_{M-1}} = aa$ et $G^i_{m_M} = bb$

$$
\begin{aligned}
u_M = & \sum_{S^P_{i,m_M}} g_{G^i_{m_M}}(H_{m_M}, S^P_{im_M}) \cdot u_{M-1} \\
= & \sum_{S^P_{i,m_M}} \mathbb{P}(S^P_{i,m_{M+1}} \mid S^P_{i,m_M}) \cdot g_{G^i_{m_M}}(H_{m_M}, S^P_{im_M}) \prod_{l=2}^{M-1} f(H_{m_{l-1}}, H_{m_l}) \times t_{G^i_{m_{M-1}}}(H_{m_{M-1}}, S^P_{i,m_M}) \\
= & \prod_{l=2}^{M-1} f(H_{m_{l-1}}, H_{m_l}) \sum_{S^P_{i,m_M}} g_{G^i_{m_M}}(H_{m_M}, S^P_{im_M}) \cdot t_{G^i_{m_{M-1}}}(H_{m_{M-1}}, S^P_{i,m_M})
\end{aligned}
$$

La table A.7 détaillant le résultat de $\sum_{S^P_{i,m_M}} g_{G^i_{m_M}}(H_{m_M}, S^P_{im_M}) \cdot t_{G^i_{m_{M-1}}}(H_{m_{M-1}}, S^P_{i,m_M})$ pour $G^i_{m_M} = aa$ et $G^i_{m_{M-1}} = bb$ permet de déterminer que

$$
\sum_{S^P_{i,m_M}} g_{G^i_{m_M}}(H_{m_M}, S^P_{im_M}) \cdot t_{G^i_{m_{M-1}}}(H_{m_{M-1}}, S^P_{i,m_M}) = f_{G^i_{m_{M-1}}, G^i_{m_M}}(H_{m_{M-1}}, H_{m_M})
$$

d'où $u_M = \prod_{k=2}^{M} f_{G^i_{m_{M-1}}, G^i_{m_M}}(H_{m_{k-1}}, H_{m_k})$

□

Après toutes les étapes présentées, la formulation de F_i devient l'équation suivante.

$$\begin{aligned}
F_i = \quad & \sum_{\texttt{Fam}_i} \Big(\prod_{k=1}^{|L|} \mathbb{P}(A^P_{\texttt{mere}(i)k}) \cdot \mathbb{P}(A^M_{\texttt{mere}(i)k}) \\
& \times \mathbb{P}(A^P_{ik} \mid A^P_{\boldsymbol{p}k}, A^M_{\boldsymbol{p}k}, S^P_{ik}) \cdot \mathbb{P}(A^M_{ik} \mid A^P_{\texttt{mere}(i)k}, A^M_{\texttt{mere}(i)k}, S^M_{ik}) \\
& \times \mathbb{P}(G^i_k \mid A^P_{ik}, A^M_{ik}) \cdot \mathbb{P}(S^P_{ik} \mid S^P_{i(k-1)}) \cdot \mathbb{P}(S^M_{ik} \mid S^M_{i(k-1)}) \Big) \\
F_i = \quad & C^i_1 \cdot C^i_2 \prod_{k \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i} f_{G^i_{m_l}, G^i_{m_k}}(H_l, H_k) \qquad , \nexists m \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i \text{ tel que } l < m < k
\end{aligned}$$

Finalement nous avons

$$\sum_{\mathbf{V} \setminus \mathcal{P}} \mathbb{P}(\mathbf{V}) = \prod_{i=1}^{n} C^i_1 \cdot C^i_2 \prod_{k \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i} f_{G^i_{m_l}, G^i_{m_k}}(H_l, H_k) \quad , \nexists m \in \mathcal{H}_{\boldsymbol{p}} \setminus \mathcal{H}_i \text{ tel que } l < m < k \tag{A.4}$$

avec

$$f_{G^i_k, G^i_l}(H_k, H_l) = \begin{cases} (1 - r_{kl}) & \text{si } (G^i_k = G^i_l \text{ et } H_k = H_l) \text{ ou } (G^i_k \neq G^i_l \text{ et } H_k \neq H_l) \\ r_{kl} & \text{si } (G^i_k = G^i_l \text{ et } (H_k \neq H_l) \text{ ou } (G^i_l \neq G^i_k \text{ et } H_l = H_k) \end{cases} \tag{A.5}$$

Résumé

Cette thèse s'articule autour de deux thèmes : la décomposition dans les modèles graphiques que sont, entre autres, les réseaux bayésiens et les réseaux de fonctions de coûts (WCSP) et la reconstruction d'haplotypes dans les pedigrees.

Nous appliquons les techniques des Wcsp pour traiter les réseaux bayésiens, en exploitant les propriétés structurelles et fonctionnelles, de manière exacte et approchée, des instances dans le cadre de l'inférence (ou d'un problème proche, celui de compter le nombre de solutions) et de l'optimisation. Nous définissons en particulier une décomposition de fonctions qui produit des fonctions portant sur un plus petit nombre de variables.

Un exemple d'application en optimisation est la reconstruction d'haplotypes. Elle est essentielle pour une meilleure prédiction de la gravité de maladie ou pour comprendre des caractères physiques particuliers. La reconstruction d'haplotypes se modélise sous forme d'un réseau bayésien. La décomposition fonctionnelle permet de réduire ce réseau bayésien en un problème d'optimisation Wcsp (Max-2SAT).

Abstract

This thesis is based on two topics : the decomposition in graphical models which are, among others, Bayesian networks and cost function networks (WCSP) and the haplotype reconstruction in pedigrees.

We apply techniques of Wcsp to treat Bayesian network. We exploit stuctural and fonctionnal properties, in an exact and approached methods. Particulary, we define a decomposition of function which produces functions with a smaller variable number.

An application example in optimization is the haplotype reconstruction. It is essential for a best prediction of seriousness of disease or to understand particular physical characters. Haplotype reconstruction is represented with a Bayesian network. The functionnal decomposition allows to reduce this Bayesian network in an optimization problem Wcsp (Max-2SAT).