Analyse combinatoire de données : structures et optimisation

Julien Darlay

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Julien DARLAY

Thèse dirigée par **Nadia BRAUNER**
et codirigée par **Julien MONCEL**

préparée au sein du **Laboratoire G-SCOP**
dans **l'école doctorale MSTII**

Analyse Combinatoire de Données
Structures et Optimisation

Thèse soutenue publiquement le **19 décembre 2011**,
devant le jury composé de :

M. Yves CRAMA
Professeur HEC Management School, Université de Liège, Rapporteur
Mme Clarisse DHAENENS
Professeur Polytech'Lille, LIFL, Rapporteur
M. Christian ARTIGUES
Chargé de recherches LAAS-CNRS Toulouse, Examinateur
M. Sylvain GRAVIER
Directeur de recherches Institut Fourier, Examinateur
Mme Nadia BRAUNER
Professeur Université Joseph Fourier, Directeur de thèse
M. Julien MONCEL
Maître de conférence Université Toulouse 1 Capitole, Co-Directeur de thèse

Remerciements

Mes remerciements s'adressent tout d'abord à Nadia Brauner et Julien Moncel qui ont accepté de diriger mes travaux de thèse.

Je souhaite ensuite remercier Yves Crama et Clarisse Dhaenens d'avoir accepté d'être les rapporteurs de cette thèse. Merci à Sylvain Gravier de m'avoir fait l'honneur de présider mon jury de thèse. Enfin je remercie doublement Christian Artigues pour avoir examiné cette thèse et pour son implication dans le challenge ROADEF.

J'adresse un remerciement tout particulier à Endre Boros, directeur du Laboratoire RUTCOR pour m'avoir encadré pendant plusieurs mois. Merci aussi aux personnels et aux doctorants de RUTCOR pour leur sympathie et l'accueil qu'ils m'ont réservé.

Je remercie également Michel Brauner, Pierre Yves Brillet ainsi que les autres médecins de l'hôpital Avicenne pour le temps qu'ils m'ont consacré.

Je voudrais exprimer ma gratitude envers mes collègues du laboratoire G-SCOP avec qui j'ai participé aux challenges ROADEF 2009 et 2010. En particulier, merci à Yann Kieffer pour ses conseils ainsi que les discussions que nous avons pu avoir. Merci enfin aux membres du laboratoire pour leur accueil, pour les midi jeux et les autres activités de l'A-DOC.

Pour conclure, je voudrais remercier ma famille et mes amis pour leur soutient constant.

Table des matières

Préambule

Les travaux présentés dans cette thèse ont pour domaines principaux l'exploration de données (*data mining*) et la recherche opérationnelle, deux disciplines à l'intersection des mathématiques et de l'informatique. L'exploration de données peut se définir comme l'apprentissage de nouvelles connaissances à partir de bases de données de taille importante par des méthodes automatiques. Ce problème peut se ramener à l'optimisation d'un programme mathématique dont l'objectif n'est que partiellement connu. Cette tâche est bien souvent difficile (au sens de la complexité). De plus elle doit être réalisée sur de grandes instances. La méthodologie de la recherche opérationnelle s'applique alors : étude de complexité, proposition de méthodes de résolutions exactes ou approchées entre autres.

En complément de cette thématique, deux projets effectués lors de cette thèse mais non détaillés dans le manuscrit consistent en la participation aux challenges ROADEF [1] 2009 et 2010.

Le challenge ROADEF 2009 a été proposé par la société Amadeus. Il concerne la gestion de perturbations dans le domaine aérien. La solution que nous [2] avons proposée consiste à établir un plan de vol pour les appareils en résolvant une série de programmes linéaires en nombres entiers puis à affecter les passagers sur ces vols en résolvant un problème de flots. Cette solution nous a permis de nous hisser à la seconde place dans la catégorie junior.

Le challenge ROADEF/EURO 2010 a été proposé par EDF et concerne la gestion de la production d'énergie. La solution proposée consiste en une recherche d'une solution réalisable à l'aide d'un algorithme dédié d'énumération pour déterminer les dates d'arrêts des centrales nucléaires et d'un algorithme glouton pour la planification de la production d'électricité. Une seconde phase améliore localement la solution en modifiant les dates d'arrêts. Cette solution nous [3] a permis de terminer à la troisième place dans la catégorie sénior.

Ce manuscrit se divise en deux grandes parties. La première, plutôt pra-

1. `http://challenge.roadef.org/`
2. avec Louis-Philippe Kronek, Susann Schrenk et Lilia Zaourar
3. avec Louis Esperet, Yann Kieffer, Guyslain Naves et Valentin Weber

tique, traite de méthodes d'exploration de données basées sur l'Analyse Combinatoire de Données (ACD) et ses extensions. Cette méthode, issue de la communauté de la recherche opérationnelle, repose sur l'optimisation combinatoire et les fonctions booléennes. La seconde partie apporte un regard plus théorique, avec l'étude de problèmes d'exploration de données vus comme des problèmes d'optimisation combinatoire.

Le Chapitre 1 est une introduction à l'exploration de données. Il présente les concepts généraux ainsi que quelques méthodes de la littérature. Ce chapitre se termine avec les liens entre l'exploration de données et la recherche opérationnelle.

Le Chapitre 2 introduit l'analyse combinatoire de données. Il présente un nouvel algorithme basé sur un algorithme exponentiel exact et rendu utilisable en pratique par des techniques d'échantillonnage. Ce travail a été réalisé en collaboration avec Endre Boros, directeur du laboratoire RUTCOR de l'université Rutgers (NJ, USA). L'analyse combinatoire de données est ensuite appliquée sur des données originales issues d'un partenariat avec l'hôpital Avicenne de Paris.

Le Chapitre 3 traite d'analyse de temps de survie. Il s'agit d'un problème d'exploration de données particulier où l'objectif est de modéliser le temps avant un événement à partir d'observations éventuellement censurées (on ne connaît qu'une information partielle sur le temps avant leur événement). Une nouvelle extension de l'Analyse Combinatoire de Données pour ces problèmes est proposée. Elle se base sur des techniques classiques de la recherche opérationnelle, entre autres la programmation linéaire en nombres entiers et les heuristiques gloutonnes. Cette méthode est ensuite comparée à d'autres sur des données de la littérature.

Le Chapitre 4 s'intéresse à un problème de partition de graphes rencontré lors de l'extension de l'ACD aux problèmes de temps de survie. Ce problème est issu de l'exploration de données et plus particulièrement de la détection de communautés. Nous commençons par une revue des méthodes de la littérature pour ces problèmes puis nous montrons que la partition d'un graphe en graphes denses est NP-difficile. Le chapitre se termine par un algorithme polynomial exact pour les arbres reposant sur la théorie des couplages.

Le Chapitre 5 concerne la sélection d'attributs. Ce problème se pose avant toute exploration de données. Il consiste à trouver les attributs les plus pertinents parmi ceux qui décrivent les données. Dans le cas où les attributs sont binaires, il généralise le problème de couverture par les tests. Nous nous intéressons à la complexité ainsi qu'à des bornes sur la solution optimale lorsque l'instance est soumise à différentes contraintes.

L'Annexe A rappelle les définitions classiques en théorie des graphes ainsi que des notions de complexité. L'Annexe B s'intéresse à un problème ouvert lié à la notion de densité étudiée dans le Chapitre 4. Il consiste à trouver le sous-graphe le plus dense avec un nombre de sommets fixés. Ce problème

généralise le problème de clique maximum. Sa complexité sur les graphes d'intervalles est ouverte depuis 1984. Cette annexe comporte les avancées récentes sur ce problème ainsi que des pistes de recherche.

Chapitre 1

Exploration de données et recherche opérationnelle

L'exploration de données est une discipline récente de l'informatique. Elle peut être définie comme l'apprentissage de concepts originaux à partir d'une grande quantité de données, par des méthodes automatisées. Les applications de ces techniques sont nombreuses comme l'exploitation de fichiers clients, l'aide à la décision, l'aide au diagnostic. Pour cela, il faut disposer d'une grande quantité de données et de la puissance de calcul permettant de les traiter.

Les progrès de l'informatique ont permis le développement de l'exploration de données. Tout d'abord, les systèmes informatiques se sont largement répandus depuis les années 1970 et sont aujourd'hui présents dans tous les domaines. D'autre part les capacités de stockage des disques durs d'ordinateurs grand public augmentent chaque année, allant de quelques mégaoctets en 1980 à plusieurs téraoctets en 2011. L'augmentation des capacités de stockage n'est cependant pas suffisante, l'augmentation de la puissance de calcul joue un rôle crucial pour analyser ces grandes quantités de données. La loi de Moore [67] stipule que la puissance de calcul des processeurs double chaque année. Cette loi a été établie en 1965 et reste vérifiée jusqu'à aujourd'hui. Enfin, les progrès faits en algorithmique forment le dernier point permettant d'expliquer l'intérêt pour l'exploration de données. Pour prendre un exemple bien connu du domaine de l'optimisation, les algorithmes permettant la résolution de programmes linéaires ont connu de nettes améliorations depuis les années 1980. Bixby [9] a montré expérimentalement que sur un grand nombre d'instances, trois ordres de grandeur sur le temps de résolution ont été gagnés, uniquement grâce aux progrès faits sur les algorithmes.

L'exploration de données s'est développée à travers de nombreux algorithmes reposant sur des paradigmes différents. Cette diversité d'algorithmes peut s'expliquer par le *no-free-lunch theorem* de Wolpert [93]. Ce théorème stipule que sans connaissance *a priori* sur les données, tous les algorithmes

d'exploration de données se valent. Nous allons voir, plus loin dans ce chapitre, comment évaluer et comparer la performance des algorithmes de manière empirique. En pratique, le choix d'un algorithme plutôt qu'un autre se fait en fonction de l'application et des données à exploiter.

Malgré une grande diversité dans les algorithmes d'exploration de données, ceux-ci suivent globalement un même schéma commun. La Section 1.1 donne les grandes catégories d'algorithmes d'apprentissage ainsi que le schéma qu'ils suivent. Nous allons voir les problèmes qui se posent lors de l'évaluation des algorithmes d'apprentissages et les solutions retenues en pratique.

Quelques domaines d'applications

Le champ d'application de l'exploration de données est vaste. Un récent sondage [1] montre que les utilisations principales de ces techniques sont variées : la gestion de relation client, le domaine bancaire, la santé, la finance et les télécoms.

L'exemple le plus fréquemment cité est certainement celui de la grande distribution. On le retrouve en introduction de nombreux ouvrages dédiés à l'exploration de données [88, 92]. L'analyse des tickets de caisse permet d'identifier des profils de clients afin de leur proposer des services spécifiques. Les données récoltées lors de la création de carte de fidélité permettent d'enrichir les profils avec des données démographiques.

En ce qui concerne le domaine bancaire, l'exploration de données est utilisée comme aide à la décision pour évaluer le risque de non-remboursement des prêts, cette activité est connue sous le nom de *credit scoring*. On retrouve ainsi de nombreux jeux de données issus du domaine bancaire sur des sites dédiés à la communauté de l'exploration de données tels que l'*UCI repository* [2].

Dans le domaine de la santé, l'exploration de données sert d'outil d'aide au diagnostic [39, 60]. La Section 2.4 propose une application originale pour la classification de patients atteints de pneumopathies. Dans ces applications, l'objectif est d'obtenir, à partir d'un grand ensemble de données, des règles simples permettant de classer les patients selon leur pathologie. Ce classement est effectué à partir de mesures biologiques et d'observations radiologiques.

Les domaines d'applications de l'exploration de données sont encore nombreux, on peut citer entre autres l'analyse de réseaux télécoms, la détection de *spam*, la reconnaissance de formes dans les images.

Avec des domaines d'applications aussi variés, les méthodes utilisées pour l'apprentissage peuvent être très différentes. En effet, les méthodes

1. `http://www.kdnuggets.com/polls/2010/analytics-data-mining-industries-applications.html`
2. `http://archive.ics.uci.edu/ml/datasets.html`

mises en place pour la reconnaissance de formes et pour l'extraction d'informations à partir de textes ne feront pas appel aux mêmes concepts. Dans cette thèse, nous allons nous intéresser aux problèmes d'apprentissage dont les données peuvent se représenter sous la forme d'un tableau. Dans ce tableau, chaque ligne décrit une observation du concept à apprendre et chaque colonne contient une information sur cette observation. À titre d'exemple, considérons une application médicale où l'objectif est de prédire si un patient est malade ou non à partir de ses symptômes. Les informations recueillies pour l'apprentissage sont présentées Tableau 1.1. Les lignes correspondent à des patients dont nous connaissons déjà les symptômes et leur statut vis-à-vis de la maladie (colonne Diagnostic). Les colonnes du tableau contiennent des informations médicales comme la présence de fièvre, l'âge des patients.

Id.	Âge	Fatigue	Fièvre	Douleur	Diagnostic
0	76	oui	oui	forte	positif
1	63	oui	non	très forte	positif
2	52	non	non	très forte	positif
3	35	non	oui	modérée	négatif
4	45	non	non	faible	négatif
5	58	non	oui	faible	négatif

Tableau 1.1 – Exemple de données médicales (fictives).

Ce chapitre présente le contexte général de l'exploration de données. La Section 1.1 présente le schéma général commun à la plupart des méthodes d'apprentissage, de la préparation des données à l'évaluation des modèles générés. La Section 1.2 introduit trois méthodes classiques reposant sur des concepts différents. Enfin, la Section 1.3 explicite les liens entre exploration de données et recherche opérationnelle.

1.1 Schéma d'apprentissage

1.1.1 Définitions

Une **observation** est un vecteur de réels représentant les informations connues sur un individu. Chaque élément d'une observation est un **attribut**. En pratique, les informations dont nous disponsons sur les observations peuvent être de plusieurs types, nous verrons dans la Section 1.1.2 comment les ramener à des vecteurs de $\mathbb{R}^m$.

Un **jeu de données** ou **base de données** est une collection d'observations. Dans certains cas d'apprentissage, les observations sont munies d'une valeur supplémentaire particulière appelée **valeur cible**. Elle représente la valeur à prédire, par exemple, le diagnostic médical. Pour certaines méthodes dont l'analyse combinatoire de données, il n'est pas possible d'apprendre un

concept si deux observations avec des attributs identiques ont des valeurs cibles différentes. On retire ces observations de la base et on se ramène à un ensemble d'observations.

Un **modèle** est une fonction qui associe à un vecteur d'attributs, une valeur dans l'espace des valeurs cibles. On trouve parfois dans la littérature le terme d'**hypothèse** [23]. Dans l'exemple du Tableau 1.1, la règle : "Si l'observation a une douleur forte ou très forte alors son diagnostic est positif, sinon il est négatif" est un modèle simple permettant d'attribuer un diagnostic à une observation en fonction de ses attributs. La valeur prédite par un modèle n'est pas nécessairement la valeur cible de l'observation, les erreurs sont autorisées même si en pratique on cherche un modèle qui commet peu d'erreurs.

1.1.2 Préparation des données

La préparation des données intervient en amont de l'apprentissage. Elle sert à transformer les données réelles en données compréhensibles par les algorithmes d'apprentissage. Nous décrivons ici les problèmes classiques rencontrés lors des expériences réalisées durant la préparation de la thèse.

Nous l'avons vu dans le Tableau 1.1, les attributs ne sont pas toujours des valeurs numériques. Une première transformation est nécessaire pour nous ramener à des observations telles qu'elles ont été définies. Nous distinguons quatre types d'attributs : booléen, numérique, catégorie ordonnée et catégorie non ordonnée. Dans l'exemple précédent, la présence de fièvre est un attribut booléen, l'âge un attribut numérique et la douleur un attribut de catégorie ordonnée. Dans la pratique, d'autres types d'attributs peuvent être présents en base : des séries temporelles, des images, du texte par exemple. Ces attributs nécessitent des traitements particuliers et nous renvoyons le lecteur au chapitre 7 de [92] pour un aperçu des techniques existantes.

Les attributs booléens ou numériques ne nécessitent pas de transformation puisqu'ils sont déjà décrits par des réels. Les attributs de catégories peuvent être remplacés par autant d'attributs booléens que de catégories. Une autre solution consiste à remplacer les catégories par des entiers. Cette solution est plus compacte et permet dans le cas où un ordre total existe entre les catégories de préserver cet ordre. Les données du Tableau 1.1 sont ainsi transformées en données numériques dans le Tableau 1.2.

Des algorithmes d'apprentissage nécessitent en entrée un jeu de données dont les attributs sont purement binaires, on parle alors de binarisation des attributs. Ces méthodes sont présentées plus tard dans la Section 2.2.1.

Parmi l'ensemble des attributs en base, il peut y avoir des informations inutiles ou redondantes. On peut supprimer ces informations pour réduire la taille de la base de données et donc diminuer le temps d'exécution de l'algorithme d'apprentissage. Ce problème est connu dans la littérature comme le problème de sélection d'attributs (*feature selection*). Nous verrons dans le

Id.	Âge	Fatigue	Fièvre	Douleur	Diagnostic
0	76	1	1	2	1
1	63	1	0	3	1
2	52	0	0	3	1
3	35	0	1	1	0
4	45	0	0	0	0
5	58	0	1	0	0

Tableau 1.2 – Exemple de données médicales avec des attributs transformés en attributs numériques.

Chapitre 5 les différentes solutions proposées dans la littérature ainsi qu'une approche combinatoire originale.

Les traitements des données présentés ci-dessus sont les traitements les plus classiques. Dans certains cas, le jeu de données peut être incomplet. Il faut dans ce cas traiter de manière particulière les valeurs manquantes. Certains algorithmes d'apprentissage les tolèrent, alors que pour d'autres il faut les compléter avec, par exemple, une estimation. D'autres algorithmes sont sensibles à la distribution des valeurs de chaque attribut, en favorisant, par exemple, les attributs ayant de grandes valeurs numériques par rapport à des attributs booléens. Pour plus d'informations sur la préparation des données, nous renvoyons le lecteur à l'ouvrage de référence [92].

1.1.3 Algorithme d'apprentissage

Un algorithme d'apprentissage est un algorithme qui, à partir d'un jeu de données, retourne un modèle. On distingue alors deux types d'algorithmes : les algorithmes d'apprentissage supervisé et les non supervisé.

Les algorithmes d'apprentissage **non supervisé** fournissent seulement des informations sur les données. Les algorithmes de partitionnement de données (*clustering*), regroupent les observations en groupe d'observations homogènes suivant des mesures de proximité. Les algorithmes de recherche de règles d'association retournent des règles sur les attributs du type : "si l'attribut i prend la valeur v_i et l'attribut j prend la valeur v_j alors l'attribut k prend la valeur v_k".

Les algorithmes d'apprentissage **supervisé** utilisent en plus des données, un attribut cible et retournent un modèle. L'exemple médical présenté dans le Tableau 1.2 est un cas d'apprentissage supervisé. On distingue différents types d'apprentissage supervisé en fonction de la nature de l'attribut cible. Lorsqu'il s'agit d'un attribut de catégorie, on parle de problème de **classification**, alors que pour une valeur numérique, on parle de problème de **régression**. Les algorithmes permettant de traiter ces deux types de problèmes sont, par nature, différents bien qu'il existe dans certains cas des extensions permettant de traiter les deux cas.

À titre d'exemple, nous allons voir un algorithme d'apprentissage supervisé simple pour un problème de classification à deux classes. Nous considérons les données du Tableau 1.2 où chaque observation est un point dans un espace à quatre dimensions. Ces dimensions sont décrites respectivement par les attributs : âge, fatigue, fièvre et douleur. Une solution pour distinguer les observations avec le diagnostic 1 (positif) et le diagnostic 0 (négatif) est de trouver l'équation d'un hyperplan séparant l'espace en deux régions : une région contenant des observations avec un diagnostic positif et une région contenant des observations avec un diagnostic négatif. Ce principe est à la base des méthodes de type SVM (*Support Vector Machine*) proposées par Vapnik [14]. Un tel hyperplan est décrit par un vecteur a et un scalaire b tels que $ax + b = 0$ avec x un vecteur de réels.

Nous souhaitons que les observations avec le diagnostic 0 soient d'un coté de l'hyperplan et les observations avec le diagnostic 1 soient de l'autre coté. En notant Ω^+ l'ensemble des observations avec le diagnostic 1 et Ω^- l'ensemble des observations avec le diagnostic 0, l'équation de l'hyperplan doit vérifier :

$$\begin{array}{rcll} ax + b & > & 0 & \forall x \in \Omega^+, \\ ay + b & < & 0 & \forall y \in \Omega^-. \end{array}$$

Mangasarian [65, 66] propose le programme linéaire suivant pour trouver l'équation d'un tel hyperplan en minimisant, par exemple, la norme 1 du vecteur a :

$$\begin{array}{lrcll} \min & \sum_i a'_i & & & \\ \text{s.t.} & ax + b & \geq & 1 & \forall x \in \Omega^+ \\ & ay + b & \leq & -1 & \forall y \in \Omega^- \\ & a'_i & \geq & a_i & \forall i \in \{0, 1, 2, 3\} \\ & a'_i & \geq & -a_i & \forall i \in \{0, 1, 2, 3\} \end{array}$$

Les variables a'_i prennent la valeur de $|a_i|$ et l'objectif du problème consiste à minimiser la norme 1 du vecteur a.

Coefficient	Valeur
$a_{\text{âge}}$	0.0317
a_{fatigue}	0
$a_{\text{fièvre}}$	0
a_{douleur}	0.730
b	-2.84

Tableau 1.3 – Coefficients de l'hyperplan séparant les observations du Tableau 1.2 (arrondis au troisième chiffre significatif).

La solution de ce problème pour les données du Tableau 1.2 est décrite dans le Tableau 1.3. L'hyperplan ainsi trouvé forme un modèle simple. Il

permet de classer une nouvelle observation à partir des observations du jeu de données, en regardant dans quelle région de l'espace elle se situe.

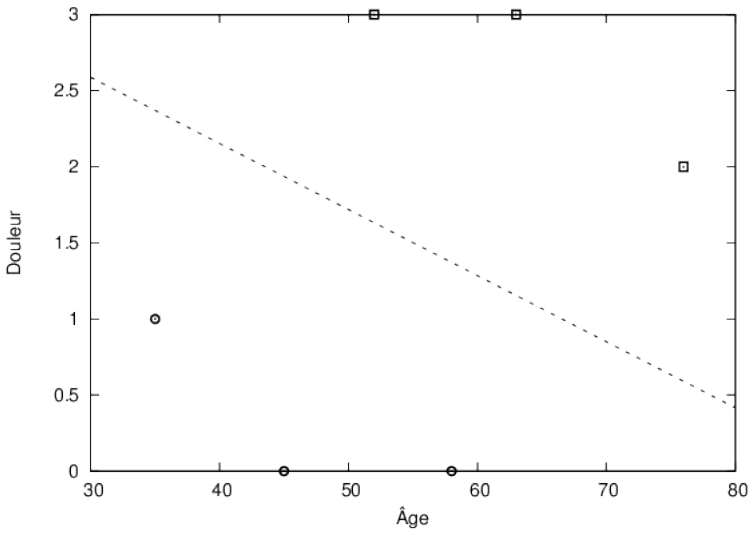

Figure 1.1 – Hyperplan séparant les observations du Tableau 1.2. Les observations avec un diagnostic positif sont représentées par des carrés, les autres par des cercles.

1.1.4 Évaluation de l'apprentissage

Nous venons de voir un exemple d'algorithme de classification supervisée qui retourne un modèle simple. En pratique, il n'est pas toujours possible de trouver un modèle simple qui ne se trompe jamais sur la base de données. Dans ce cas, il faut pouvoir évaluer les performances du modèle. Nous allons voir les indicateurs de performances classiques pour évaluer un modèle dans le cas d'un problème de classification supervisée à deux classes. Des indicateurs existent aussi pour les autres types de problèmes, nous renvoyons le lecteur aux ouvrages de références [88, 92].

Dans les problèmes à deux classes, lorsque le modèle attribue toujours une classe à une observation, il ne peut y avoir que quatre situations, résumées dans le Tableau 1.4. On observe alors que :

- les **Vrais Positifs** (VP) sont les observations de la classe 1 qui sont bien classées par le modèle ;
- les **Vrais Négatifs** (VN) sont les observations de la classe 0 qui sont bien classées par le modèle ;
- les **Faux Positifs** (FP) sont les observations de la classe 0 qui sont mal classées par le modèle ;
- les **Faux Négatifs** (FN) sont les observations de la classe 1 qui sont mal classées par le modèle.

	Prédiction du modèle	
	1	0
Classe 1	Vrai Positif	Faux Négatif
Classe 0	Faux Positif	Vrai Négatif

Tableau 1.4 – Différentes prédictions pour un problème de classification à deux classes.

Les faux positifs et les faux négatifs correspondent au nombre d'erreurs commises par le modèle sur chacune des deux classes. Pour mesurer le taux d'observations bien classées, on utilise la **précision** (*accuracy*) définie par :

$$acc = \frac{VP + VN}{VP + VN + FP + FN}.$$

Il existe d'autres critères de performance pour les modèles générés. On retrouve ainsi en médecine la **sensibilité** et la **spécificité** données respectivement par :

$$se = \frac{VP}{VP + FN},\ sp = \frac{VN}{VN + FP}.$$

Ces deux critères mesurent la capacité à classer correctement les observations de chaque classe. La qualité des prédictions du modèle peut être évaluée par la **valeur prédictive positive** et la **valeur prédictive négative** données respectivement par :

$$vpp = \frac{VP}{VP + FP},\ vpn = \frac{VN}{VN + FN}.$$

Les critères présentés précédemment sont appliqués à la **base d'entraînement** (l'ensemble des observations ayant servi à l'apprentissage). On parle alors de performance en entraînement. On peut reprocher à cette évaluation d'être biaisée puisque les données qui servent à l'apprentissage servent aussi à l'évaluation. En pratique, on mesure les performances du modèle sur une base d'observations inconnues de l'algorithme d'apprentissage. Cette base est appelée **base de test** et permet de mesurer la capacité de l'algorithme à généraliser les concepts appris sur de nouvelles observations. On parle alors de performance en généralisation.

Une base de test est malheureusement difficile à obtenir en pratique. Effectuer une étude pour obtenir de nouvelles données coûte du temps et de l'argent. Pour éviter cela, on partitionne la base de données en une base d'entraînement et une base de test, on parle alors de **validation croisée**. Cette partition est aléatoire mais conserve les proportions d'observations de chaque classe. Pour s'affranchir du choix aléatoire de la base d'entraînement et de test, on utilise la technique des *k-folds*. La base est partitionnée en k classes étiquetées de 1 à k et l'algorithme d'apprentissage est exécuté k

fois. À l'exécution i, les classes avec une étiquette différente de i servent de base d'entraînement alors que la classe d'étiquette i sert de base de test. La performance de l'algorithme est donc la moyenne des performances sur chacune des exécutions. La Figure 1.2 présente un 5-*fold*.

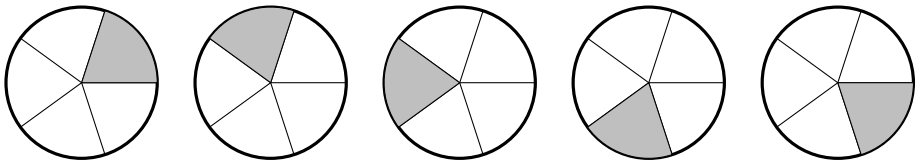

Figure 1.2 – Un 5-*fold*, chaque disque représente une exécution de l'algorithme avec en blanc les parties de la base servant à l'apprentissage, en gris la partie servant au test.

L'utilisation d'une base de test permet le contrôle du **sur-apprentissage** (*overfitting*). Ce phénomène apparaît lorsque l'algorithme génère des modèles complexes pour améliorer ses performances sur la base d'entraînement. Cette complexité est due à l'apprentissage d'erreurs, d'imprécisions dans les données. Les performances sur la base de test deviennent en principe mauvaises.

Pour illustrer ce phénomène considérons un jeu de données où les observations sont en théorie linéairement séparables en deux classes. À cause d'imprécisions dans la base de données, les données ne sont plus séparables par un plan sans commettre des erreurs. Deux solutions se distinguent, garder une séparatrice linéaire en commettant des erreurs ou utiliser une séparatrice plus complexe. Ces deux situations sont illustrées Figure 1.3. La courbe complexe ne commet pas d'erreur sur la base d'apprentissage alors que le plan en commet trois. Sur une base de test, la courbe complexe aura tendance à commettre plus d'erreurs que le plan puisque nous avons supposé que les données étaient linéairement séparables.

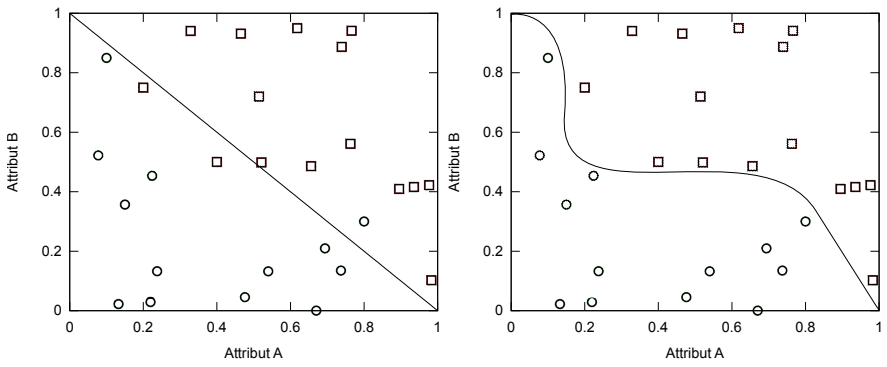

Figure 1.3 – Deux courbes séparatrices d'un jeu de données bruité.

Le phénomène de sur-apprentissage fait apparaître la notion de complexité d'un modèle. La Figure 1.4 met en évidence le lien entre complexité du modèle et ses performances. Ces courbes sont issues d'une expérience avec un algorithme d'apprentissage dont la complexité dépend du nombre d'itérations qu'il effectue. La courbe en trait plein représente la précision du modèle sur la base d'entraînement et la courbe en trait pointillé la précision du modèle pour les données de tests. Les performances sur la base d'entraînement augmentent avec la complexité du modèle alors que les performances sur la base de test augmentent au début puis diminuent au-delà de la vingtième itération. C'est à ce moment que ce produit le sur-apprentissage, lorsque les performances sur la base de test diminuent.

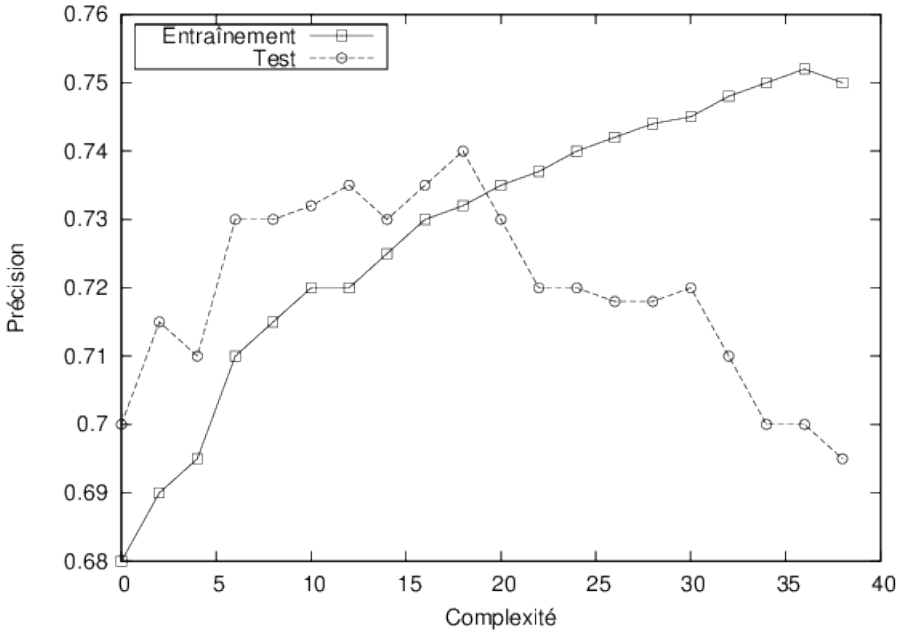

Figure 1.4 – Illustration du phénomène de sur-apprentissage.

En conclusion, l'évaluation d'un modèle généré par un algorithme d'apprentissage peut se faire grâce à plusieurs critères. Ils doivent être appliqués sur des données externes pour ne pas être biaisés par du sur-apprentissage. Le contrôle du sur-apprentissage se fait donc avec la validation croisée, ou en limitant la complexité du modèle généré (en utilisant un plan à la place d'une courbe sur l'exemple de la Figure 1.3).

1.2 Algorithmes classiques

Nous allons voir dans cette section des algorithmes classiques d'apprentissage. Ces algorithmes utilisent des techniques de recherche opérationnelle.

Nous commençons par deux algorithmes d'apprentissage supervisé : les machines à vecteurs de support (SVM) et les arbres de décision. Les SVM ont été évoquées dans la Section 1.1.3. Les arbres de décision génèrent des modèles sous forme d'arbre, facilement interprétables. Enfin, le troisième algorithme présenté est un algorithme d'apprentissage non supervisé.

1.2.1 Machine à vecteurs de support

Le principe des machines à vecteurs de support (SVM) [14] a été expliqué rapidement en introduisant un premier algorithme d'apprentissage dans la Section 1.1.3. Ces méthodes ont suscité un grand intérêt car elles reposent sur un contexte théorique fort [90] et des algorithmes d'optimisation linéaire ou quadratique particulièrement efficaces. De plus, l'efficacité de ces algorithmes d'optimisation permet de traiter des problèmes avec un grand nombre d'attributs.

L'idée générale des SVM est de rechercher un hyperplan permettant de séparer les données en fonction de leur classe dans l'espace des attributs. Cet hyperplan doit maximiser la marge, c'est-à-dire la plus petite distance à une observation. La maximisation de la marge est justifiée par la théorie statistique de l'apprentissage [90].

Lorsque les données ne sont pas linéairement sépérables, une solution consiste à introduire des dimensions supplémentaires. Par exemple, dans un espace à deux dimensions x et y, des observations séparées par un cercle d'équation $x^2 + y^2 = 1$ ne peuvent pas être séparées par un plan. Elles sont par contre séparables dans un espace à trois dimensions où la nouvelle dimension z est définie par $z = x^2 + y^2$. Cette situation est illustrée Figure 1.5.

Les SVM constituent un ensemble de méthodes basées sur la résolution de problèmes d'optimisation linéaires ou quadratiques. Ces méthodes sont détaillées dans des ouvrages classiques dédiés à l'analyse de données [23, 92]. On retrouve parmi ces méthodes des techniques classiques de recherche opérationnelle comme la programmation linéaire [15, 66].

1.2.2 Arbres de décision

Les arbres de décision [92] sont des algorithmes de classification qui fournissent des modèles sous forme d'arbre. Les nœuds internes de l'arbre correspondent à des tests sur les attributs alors que les feuilles sont des valeurs cibles. La Figure 1.6 donne un arbre de décision pour les données du Tableau 1.2.

La construction d'un arbre de décision pour un jeu de données Ω se fait récursivement. Si Ω ne contient que des observations appartenant à la même classe alors l'algorithme retourne une feuille avec cette classe comme étiquette. Si Ω contient des observations appartenant à des classes différentes alors il est partitionné en sous-ensembles selon un test sur les attributs de

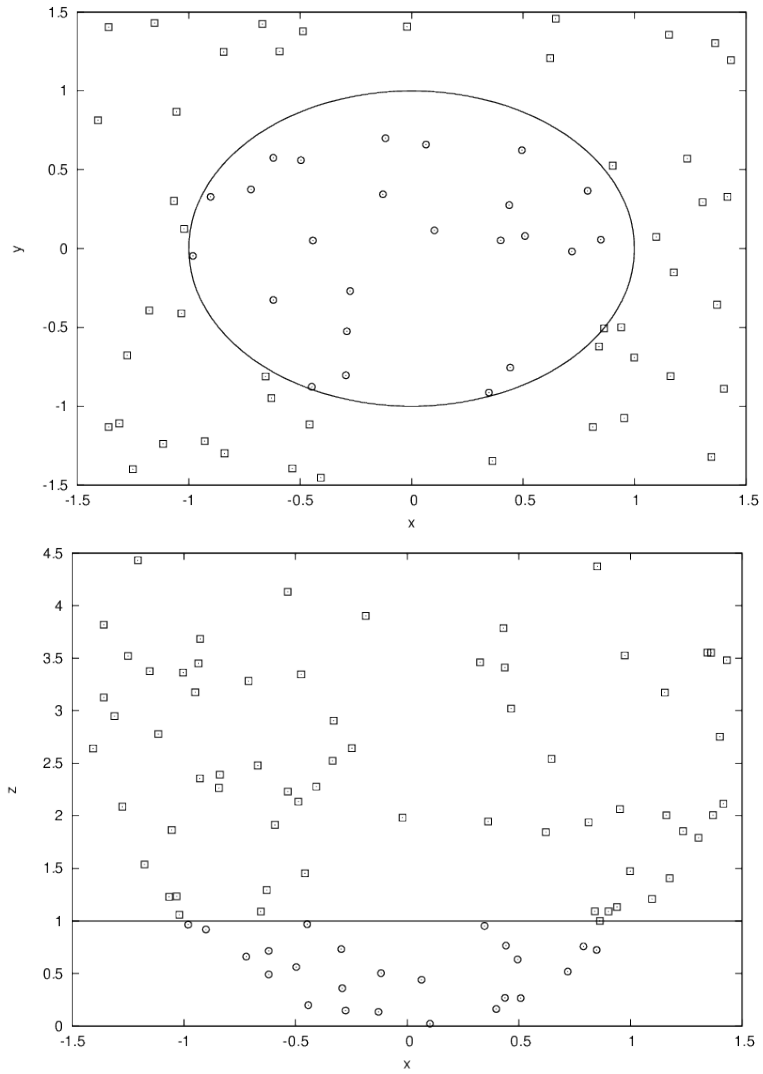

Figure 1.5 – Exemple de données non séparables linéairement (figure du haut) et leur projection dans un espace de dimension plus grand (figure du bas).

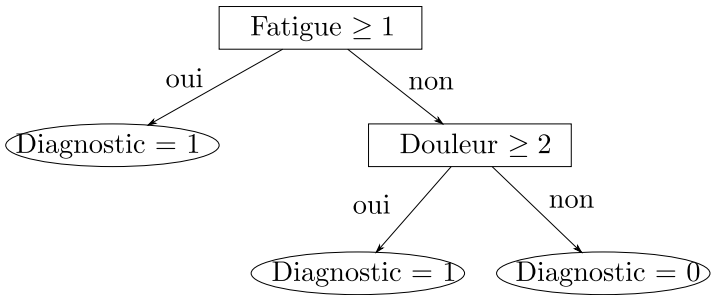

Figure 1.6 – Un arbre de décision pour les données du Tableau 1.2.

Ω. Un arbre de décision est construit pour chaque sous-ensemble. Ces arbres sont alors reliés au nœud du test permettant de partitionner Ω. Pour éviter le sur-apprentissage, une phase d'élagage de l'arbre a lieu pendant ou après la construction de l'arbre. Son rôle consiste à supprimer certains sous-arbres et à les remplacer par des feuilles dont les observations sont majoritairement de la même classe.

Il existe dans la littérature plusieurs algorithmes permettant de générer des arbres de décisions : ID3 [77], C4.5 [78] et CART [17]. Ces algorithmes diffèrent sur leur manière de sélectionner le test permettant la séparation des données et la façon d'élaguer l'arbre.

Le principal avantage des modèles à base d'arbre de décision est leur simplicité d'interprétation. Un arbre de décision est plus facilement lisible que l'équation d'un hyperplan retournée par un algorithme de type SVM.

1.2.3 *k-means*

L'algorithme des *k-means* [64] est un algorithme d'apprentissage non supervisé. L'algorithme retourne une partition du jeu de données en k classes, contenant des observations avec des attributs proches. Cette partition est obtenue à partir de k centres définissant les classes. Les observations sont affectées aux classes de façon à minimiser la somme des distances euclidiennes des observations au centre de leur classe. Ce problème étant NP-difficile [4, 28], on utilise en pratique une heuristique pour le résoudre.

L'algorithme classique commence par tirer k centres au hasard. Les observations sont alors partitionnées en fonction de ces k centres. Le barycentre des observations de chaque classe définit le nouveau centre et la procédure recommence avec ces k nouveaux centres. L'algorithme s'arrête lorsqu'un critère d'arrêt est satisfait, typiquement lorsqu'un nombre d'itérations est atteint ou lorsque les affectations ne changent plus.

1.3 Interactions avec la recherche opérationnelle

Les sections précédentes ont présenté l'exploration de données dans sa généralité. À travers les différents exemples, nous avons pu voir le rôle que peut jouer la recherche opérationnelle dans ce domaine. Dans cette section, nous allons voir que la plupart des problèmes d'exploration de données supervisés peuvent se voir comme des problèmes d'optimisation. Dans un second temps, nous verrons que les méthodes d'exploration de données peuvent aussi servir à concevoir des heuristiques pour la recherche opérationnelle.

1.3.1 Formalisme théorique de l'exploration de données

Avec le développement de l'exploration de données, des formalismes théoriques ont été créés [23], en particulier avec les travaux de Vapnik [90]. Dans ce cadre, un algorithme d'exploration de données recherche un modèle ou une hypothèse (pour reprendre la terminologie de la bibliographie) h parmi un ensemble d'hypothèses $\mathcal{H}$. Dans les exemples précédents, $\mathcal{H}$ est l'ensemble des hyperplans de l'espace des attributs dans le cadre des SVM, l'ensemble des arbres de décisions pour la méthode du même nom. Les observations sont munies d'une distribution de probabilité $\mathcal{D}_{\mathcal{Z}}$ sur l'ensemble des attributs $\mathcal{Z}$ et d'une classe notée c attribuée par une fonction cachée. L'hypothèse est évaluée avec une **fonction de perte** (*loss function*) l qui prend la valeur 1 si l'observation est mal classée par h et la valeur 0 sinon. Le **risque réel** R de l'hypothèse h est la valeur moyenne de la fonction de perte selon la distribution $\mathcal{D}_{\mathcal{Z}}$. Un problème d'apprentissage est modélisé comme le problème d'optimisation consistant à trouver l'hypothèse h dans l'ensemble $\mathcal{H}$ qui minimise le risque réel.

En général, la distribution $\mathcal{D}_{\mathcal{Z}}$ est inconnue et on ne peut pas calculer directement le risque réel. Le risque est alors mesuré sur les données d'entraînement, on parle dans ce cas de **risque empirique**. Le risque réel peut être borné par le risque empirique plus un terme dépendant des propriétés de $\mathcal{H}$ [23]. En pratique on peut donc trouver une hypothèse h en minimisant le risque empirique plus un terme correcteur dépendant de $\mathcal{H}$ [5, 90]. Une autre solution consiste à ajouter des contraintes sur l'espace des hypothèses $\mathcal{H}$ et à minimiser le risque empirique. Cette dernière approche est très souvent utilisée en pratique, en validant les résultats sur la base de test.

La théorie de l'exploration de données permet de modéliser la plupart des problèmes d'apprentissage supervisé en des problèmes d'optimisation. Malheureusement, il n'est pas possible d'évaluer directement la fonction objectif mais il est tout de même possible de l'approcher, en l'évaluant sur les données d'entraînement. On se retrouve alors avec un problème d'optimisation de grande taille (attributs et observations) pour lequel il est possible d'appliquer les techniques classiques de recherche opérationnelle.

1.3.2 Heuristiques à base d'exploration de données

Nous avons vu tout au long de ce chapitre que les méthodes de la recherche opérationnelle pouvaient être utilisées pour résoudre des problèmes d'analyse de données. Il existe dans la littérature des travaux où les méthodes d'exploration de données sont utilisées pour concevoir des heuristiques d'optimisation. Olafsson *et al.* [73] proposent un *survey* sur les interactions entre les deux disciplines.

Dans [59], les auteurs proposent d'utiliser un algorithme de classification dans un contexte de simulation-optimisation. On cherche à optimiser un système soumis à des phénomènes stochastiques en agissant sur des variables de décision. Évaluer une solution nécessite d'effectuer un grand nombre de simulations pour obtenir une valeur moyenne pertinente. Pour éviter de simuler des solutions qui auront de mauvaises performances, un algorithme de classification est entraîné pour reconnaître les mauvaises solutions à partir de simulations déjà effectuées.

Dans [61], les auteurs proposent de construire une heuristique pour un problème d'ordonnancement à partir de règles apprises sur des solutions existantes. La même idée est appliquée dans [55], les règles, générées par des algorithmes génétiques, sont apprises à partir de solutions optimales. Les conclusions des deux articles sont assez similaires : avec une bonne préparation des données, les heuristiques générées ont des performances comparables aux heuristiques d'ordonnancement classiques.

Nous avons vu dans les parties précédentes que les techniques de recherche opérationnelle sont très présentes dans les méthodes d'analyse de données. Quelques exemples de la littérature montrent qu'il existe, en plus faible proportion, des applications d'algorithmes d'exploration de données pour résoudre des problèmes de recherche opérationnelle.

1.4 Conclusion

Ce chapitre présente l'exploration de données, son fonctionnement et quelques algorithmes utilisés en pratique. La Section 1.1 donne les définitions de l'exploration de données et présente les différentes étapes, depuis la préparation des observations jusqu'à l'évaluation des modèles générés. La Section 1.2 présente trois algorithmes classiques d'apprentissage : les SVM, les arbres de décisions et les *k-means*. Ces trois méthodes sont présentes dans les différents chapitres de cette thèse. Enfin dans la Section 1.3, nous avons vu les liens entre exploration de données et recherche opérationnelle. En particulier, nous avons vu que les problèmes d'apprentissage peuvent se voir comme des problèmes d'optimisation avec un objectif inconnu mais qui peut être approché.

La première partie de cette thèse utilise les techniques de résolution de la recherche opérationnelle pour des problèmes d'exploration de données. Le

Chapitre 2 s'intéresse à l'Analyse Combinatoire de Données, une méthode issue de la communauté de l'optimisation discrète. Dans ce chapitre, un algorithme est proposé ainsi qu'une application originale issue de la recherche médicale. Le Chapitre 3 présente une extension de l'analyse combinatoire de données pour des problèmes d'analyse de temps de survie. L'analyse de temps de survie est une généralisation des problèmes de régression dont les applications sont essentiellement en médecine.

La seconde partie de cette thèse contient des résultats théoriques sur la complexité de certains problèmes d'exploration de données. Le Chapitre 4 traite d'un problème d'apprentissage non supervisé. Il s'agit d'un problème de partition en graphes denses pour lequel nous montrons la complexité dans le cas général. Un algorithme polynomial est donné pour un cas particulier. Le Chapitre 5 concerne la problématique de la sélection d'attributs binaires avec un regard "optimisation combinatoire". Nous rapprochons ce problème du problème des tests groupés et des codes identifiants. Nous nous intéressons alors à la complexité du problème ainsi qu'à des bornes sur les solutions optimales.

L'Annexe A rappelle les définitions classiques de la théorie des graphes ainsi que des notions de complexité. L'Annexe B concerne un problème ouvert de théorie des graphes, lié à la notion de densité évoquée dans le Chapitre 4. Ce problème consiste à trouver le sous-graphe le plus dense avec un nombre fixé de sommets. Il généralise le problème de la clique maximum, mais sa complexité sur les graphes d'intervalles est encore ouverte. Cette annexe présente les derniers résultats de la littérature, ainsi que des pistes de recherche.

Chapitre 2

Analyse combinatoire de données

Le chapitre précédent est une introduction à l'exploration de données et présente des algorithmes classiques. Dans ce chapitre, nous allons voir une méthode issue de la communauté de l'optimisation discrète : l'analyse combinatoire de données (ACD). Cette méthode repose sur la théorie des fonctions booléennes et leurs extensions. Elle s'applique à des problèmes de classification binaire avec des données binaires.

La Section 2.1 donne les définitions classiques des fonctions booléennes et de l'analyse combinatoire de données. La Section 2.2 présente les différentes méthodes de la littérature pour obtenir des modèles de l'ACD. La Section 2.3 présente un nouvel algorithme pour l'ACD reposant sur la dualisation de fonctions booléennes et l'échantillonnage aléatoire. Ce travail a été réalisé lors de mon séjour dans le laboratoire RUTCOR. La Section 2.4 rapporte les résultats d'une application médicale réalisée avec le service de radiologie de l'hôpital Avicenne.

2.1 Notations et définitions

Les définitions présentées dans cette section sont les définitions classiques de l'Analyse Combinatoire de Données[1], aussi connue sous l'acronyme anglais LAD pour *Logical Analysis of Data* [12, 24, 42].

On note O une observation présente dans la base de données et n le nombres d'observations de la base. L'ACD s'appliquant à des problèmes de classification binaire sur des données binaires, on suppose que chaque observation possède une classe $C \in \{0, 1\}$. On note Ω^+ les observations de la classe 1 et Ω^- les observations de la classe 0. Chaque observation est décrite par un vecteur de m attributs binaires $Z \in \{0, 1\}^m$. L'ACD ne tient

1. Le travail présenté dans la Section 2.3 est une collaboration avec Endre Boros, professeur et directeur du laboratoire RUTCOR (NJ,USA).

pas compte des observations dupliquées, on peut donc considérer dans la suite que Ω^+ et Ω^- sont des ensembles.

Une fonction booléenne est une fonction de $\{0,1\}^m$ dans $\{0,1\}$. Un **littéral** est constitué d'une variable binaire x ou de son complément $\bar{x}$. Une **conjonction** élémentaire est une expression booléenne constituée de littéraux reliés par le "ET" logique (noté $\wedge$). Une **disjonction** élémentaire est une expression booléenne constituée de littéraux reliés par le "OU" logique (noté $\vee$).

L'ensemble $\Omega = \Omega^+ \cup \Omega^-$ et l'ensemble $\mathcal{C}$ des classes décrivent une fonction booléenne partiellement définie [25] $f : \Omega \to \mathcal{C}$. Une telle fonction existe si et seulement si $\Omega^+ \cap \Omega^- = \emptyset$. Une extension de la fonction booléenne partiellement définie f est une fonction $g : \{0,1\}^m \to \{0,1\}$ telle que $g(O) = f(O)$ pour toute observation $O \in \Omega$.

Le Tableau 2.1 est un exemple d'un jeu de données binaires ; il est extrait de la base de données "Iris de Fisher" [33]. La base contient des observations de trois espèces d'iris décrites par la longueur et la largeur de leur sépale et de leur pétale. Il est initialement constitué de 50 observations, de 4 attributs numériques et de 3 classes. Dans cet exemple $\Omega^+ = \{O_1, O_2, O_3\}$ et $\Omega^- = \{O_4, O_5, O_6\}$; chaque observation est composée de 9 attributs binaires (z_0 jusqu'à z_8) et d'une classe $C \in \{0,1\}$.

Ω	longueur sépale				largeur sépale			long. pétale	larg. pétale	espèce Setosa
	≥ 5	≥ 5.4	≥ 6	≥ 7	≥ 3	≥ 3.25	≥ 3.5	≥ 3	≥ 1	
	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	C
O_1	1	0	0	0	1	1	1	0	0	1
O_2	1	1	0	0	1	1	1	0	0	1
O_3	0	0	0	0	1	0	0	0	0	1
O_4	1	1	1	1	1	0	0	1	1	0
O_5	1	1	0	0	1	0	0	1	1	0
O_6	1	1	1	0	1	0	0	1	1	0

Tableau 2.1 – Données "Iris" avec des attributs binaires.

L'ACD, comme d'autres méthodes d'analyse de données, fait l'hypothèse qu'il existe une fonction cachée qui associe à un vecteur d'attributs une classe. En particulier, l'ACD cherche une description simple de cette fonction cachée par une disjonction de conjonctions particulières nommées **motifs** (*patterns*). Ces conjonctions définissent des sous-espaces de l'espace des attributs $\{0,1\}^m$.

Définition 2.1. Un **motif** (*pattern*) est une conjonction élémentaire.

Une observation est **couverte** par un motif si ses attributs vérifient la conjonction du motif. Un motif **positif** couvre des observations appartenant majoritairement à la classe 1 alors qu'un motif **négatif** couvre des observa-

tions appartenant majoritairement à la classe 0. Un motif est **pur** lorsqu'il ne couvre que des observations de la même classe.

La **couverture** d'un motif P (notée $cov(P)$) est l'ensemble des observations couvertes par ce motif. Puisqu'un motif peut couvrir des observations des deux classes, on distingue la couverture positive (notée $cov^+(P)$) de la couverture négative (notée $cov^-(P)$).

En reprenant l'exemple du Tableau 2.1, la conjonction $P_0 = z_0 \wedge \bar{z}_2 \wedge z_5$ décrit un motif positif qui couvre les observations O_1, O_2. Ces observations appartenant toutes à la classe 1, le motif P_0 est un motif pur. Le motif P_0 peut se lire : "si la longueur du sépale est supérieure à 5 *et* si la longueur du sépale est inférieure à 6 *et* si la largeur du sépale est supérieure à 3.25 *alors* l'observation est de la classe Setosa".

La condition de pureté des motifs donne une garantie empirique sur l'interprétation du motif, il n'existe alors pas de contre-exemple dans la base de donnée. En pratique une telle condition est difficile à obtenir sans avoir un motif trop long et difficilement interprétable pour l'utilisateur. Des critères tels que le **degré**, l'**homogénéité** et la **prévalence** ont été proposés pour caractériser les motifs.

Définition 2.2. Le **degré** d'un motif est le nombre de littéraux dans sa conjonction.

Les définitions suivantes dépendent du type de motif (positif ou négatif). Elles sont données dans le cas des motifs positifs mais peuvent être généralisées aux motifs négatifs en remplaçant les symboles + par des symboles − dans les définitions.

Définition 2.3. L'**homogénéité** (notée h) d'un motif positif P est le rapport entre le nombre d'observations positives couvertes et le nombre d'observations couvertes :

$$h = \frac{|cov^+(P)|}{|cov(P)|}$$

Définition 2.4. La **prévalence** (notée p) d'un motif positif P représente la proportion d'observations positives couvertes par rapport au nombre d'observations positives :

$$p = \frac{|cov^+(P)|}{|\Omega^+|}$$

En continuant l'exemple du Tableau 2.1, le motif négatif $P_1 = z_0 \wedge z_1$ est de degré $d = 2$ et d'homogénéité $h = \frac{3}{4}$. La prévalence du motif positif $P_0 = z_0 \wedge \bar{z}_2 \wedge z_5$ est $p = \frac{2}{3}$.

L'objectif de l'ACD est de générer un nombre minimal de motifs (éventuellement purs) permettant de couvrir toutes les observations du jeu de données. Un ensemble de motifs couvre une observation si au moins l'un

d'entre eux la couvre. La Section 2.2 présente quelques méthodes de la littérature pour la génération de motifs. La Section 2.3 présente un algorithme pour énumérer des motifs en utilisant des résultats sur les fonctions booléennes.

On appelle **théorie positive** (notée $\mathcal{P}$) un ensemble de motifs positifs couvrant les observations de la classe 1. De manière symétrique, on peut définir une **théorie négative** (notée $\mathcal{N}$). Les motifs $P = z_5$ et $P' = \bar{z}_0 \wedge \bar{z}_1$ forment une théorie positive alors que le motif négatif $N = z_0 \wedge \bar{z}_5$ forme une théorie négative pour les données du Tableau 2.1.

Pour attribuer une classe à une observation, Boros *et al* [12] proposent d'utiliser une **fonction discriminante**. Il s'agit d'une fonction Δ qui à partir d'un vecteur d'attributs $Z \in \{0,1\}^m$ donne une valeur réelle :

$$\Delta(Z) = \sum_{P \in \mathcal{P}} P(Z) - \sum_{N \in \mathcal{N}} N(Z)$$

Le signe de cette fonction permet de déterminer si l'observation avec le vecteur d'attributs Z est couverte majoritairement par des motifs positifs ou par des motifs négatifs. Des schémas de pondération sur les motifs peuvent être appliqués en cas de déséquilibre entre le nombre de motifs positifs et le nombre de motifs négatifs. Les motifs sont ainsi vus comme des arguments en faveur de leur classe. Un discriminant positif indique que l'observation est plus proche de la classe 1 que de la classe 0. Il est parfois plus avantageux de n'attribuer une classe que lorsque la valeur absolue du discriminant dépasse un seuil et de laisser l'observation non classée sinon [5].

Définition 2.5. Un **modèle** (au sens de l'ACD) est l'association d'une théorie positive, d'une théorie négative et d'une fonction discriminante.

Dans la suite, on utilisera le terme modèle pour désigner un modèle de l'ACD ainsi que les motifs contenus dans les théories.

En pratique, les méthodes d'ACD vont générer des motifs simples (de faible degré), avec de bonnes propriétés (homogénéité et prévalence proches de 1). L'objectif est de former un modèle simple (avec peu de motifs) avec de bonnes performances sur les données d'entraînement et sur les données de test (capacité de généralisation).

2.2 Méthodes de la littérature

Les méthodes d'analyse combinatoire de données reposent presque toutes sur un même schéma. Tout d'abord il est nécessaire de préparer les données ; puisque l'ACD repose sur des techniques booléennes, il faut que les attributs soient booléens. L'étape suivante consiste à générer des motifs à partir de la base d'entraînement. Enfin la dernière étape construit un modèle en

utilisant les motifs précédemment générés. Nous allons voir pour chacune de ces étapes les méthodes classiques de la littérature.

2.2.1 Préparation des données

De manière générale avant d'appliquer les méthodes de génération de motifs et de modèles, il est nécessaire de préparer les données. Cette préparation se retrouve avant toute analyse, quelque soit la méthode. Le Chapitre 7 du livre de Witten *et al.* [92] donne les méthodes classiques pour discrétiser les données, sélectionner les attributs etc. Le cadre d'utilisation de l'ACD est particulier puisqu'il nécessite des données binaires et des observations divisées en deux classes. Des modèles d'optimisations propres ont été développés pour transformer les données en données utilisables par l'ACD.

La première étape consiste à transformer les attributs en attributs binaires. Boros *et al.* [12] proposent des méthodes pour transformer des attributs nominaux (ex : rond, carré, triangle) ou ordonnés (ex : faible, moyen, fort) en attributs binaires. Pour des attributs nominaux, un nouvel attribut binaire est introduit pour chaque élément de la catégorie. Dans le cas spécial où l'attribut ne peut prendre que deux valeurs, il est remplacé par une seule variable binaire.

Chaque attribut ordonné pouvant prendre v valeurs est remplacé par $v-1$ ou v attributs binaires représentant les points de coupe. En reprenant l'exemple faible, moyen, fort, les nouveaux attributs correspondent à "$\leq$ faible", "$\leq$ moyen" et "$\leq$ fort". On peut remarquer que seuls les deux derniers attributs sont nécessaires, puisque si ils valent tous les deux 0, le premier est forcément à 1. En pratique il peut être nécessaire d'introduire cette redondance puisqu'on cherche à générer des modèles de faible degré. Si on ne garde que les deux derniers attributs alors pour exprimer que l'attribut original vaut "$\leq$ faible" il faut deux attributs binaires dans le motif.

Les cas des attributs continus a aussi été traité dans [11, 12]. L'idée générale est d'introduire des points de coupe parmi les différentes valeurs que peuvent prendre les attributs. En utilisant de manière naïve tous les points de coupe, on peut se retrouver à introduire n variables binaires (n étant le nombre d'observations) dans le cas où chaque observation a une valeur différente. Boros *et al.* [11] ont étudié de manière théorique le problème d'optimisation consistant à sélectionner le nombre minimal de points de coupe. Ce problème est NP-difficile mais peut être résolu en pratique par des méthodes classiques de programmation linéaire en nombres entiers [12] ou par des algorithmes polynomiaux dans certains cas particuliers [11].

Une fois que les données sont binaires, il est possible que le nombre d'attributs soit grand. Sélectionner un sous-ensemble minimal d'attributs tout en gardant la distinction entre observations positives et négatives est un problème connu en analyse de données sous le nom de sélection d'attributs

(*feature selection* [92]). Une fois de plus, l'ACD offre un cadre particulier avec ses données binaires. On parle alors de sélection de support [24].

Un **support** S est un sous-ensemble minimal d'attributs tel que les restrictions de Ω^+ et Ω^- à S soient encore disjointes. Le problème d'optimisation consistant à minimiser la cardinalité de S est NP-difficile [13]. Une modélisation sous forme de problème de couverture minimale d'ensembles (problème SP5 de [35]) permet de proposer des solutions en pratique lorsque le nombre d'attributs n'est pas trop grand pour que le problème puisse être résolu par des algorithmes de type *branch & bound*.

L'ACD est une méthode de classification binaire. Lorsque les observations appartiennent à plus de deux classes, le problème mutliclasse est transformé en plusieurs problèmes de classification binaire. En pratique, la solution la plus simple consiste à transformer un problème à $|\mathcal{C}|$ classes en $|\mathcal{C}|$ problèmes du type une classe contre les autres. Une méthode plus évoluée transforme le problème multiclasse en plusieurs problèmes binaires de type plusieurs classes contre plusieurs autres. Cette méthode est connue sous le nom de ECOC pour *Error Correcting Output Code* [32].

2.2.2 Génération de motifs

La génération de motifs est l'étape suivante de l'ACD lorsque les données ont été transformées en données binaires. Il existe deux grandes familles d'algorithmes pour générer des motifs : les méthodes basées sur l'énumération et celles reposant sur des problèmes d'optimisation. Nous allons voir dans cette section quelques méthodes de la littérature utilisant ces deux approches.

Les premières méthodes exhaustives [12] cherchent à générer des motifs purs en étant guidées par deux objectifs : trouver des motifs de degré faible et couvrir chaque observation par au minimum un motif. Pour satisfaire le premier objectif un algorithme *bottom-up* est proposé, il consiste à partir d'un motif vide à ajouter des littéraux au fur et à mesure jusqu'à obtenir un motif satisfaisant des contraintes d'homogénéité, de prévalence, etc. Le nombre de motifs de degré d sur un jeu de données de m attributs est $2^d\binom{m}{d}$. En pratique cette méthode est utilisée pour trouver des motifs avec un degré maximal de 3 ou 4. Le logiciel Ladoscope [1] propose une implémentation de cet algorithme. Il est possible que des motifs de degrés 3 ou 4 ne puissent pas couvrir toutes les observations ; pour cela une approche *top-down* est proposée en complément. Elle consiste à créer n motifs de degré m caractérisant les n observations en base. Puis elle va créer des nouveaux motifs à partir des anciens en retirant des littéraux. La méthode s'arrête lorsqu'elle ne peut plus générer de motifs sans violer des contraintes sur l'homogénéité, la prévalence ou la couverture.

1. `http://pit.kamick.free.fr/lemaire/software-lad.html`

Alexe et Hammer [2] proposent un algorithme de type *top-down* pour la génération de motifs étendus (*spanned*). Les motifs étendus sont des motifs pour lesquels il n'est plus possible d'ajouter de littéral. L'intérêt des motifs étendus est discuté dans [1], ils commettent moins d'erreurs de généralisation (sur de nouvelles données) mais peuvent laisser des observations non classées. Les motifs premiers (*prime*) sont les motifs pour lesquels il n'est plus possible d'enlever de littéral à son expression booléenne. Un algorithme de génération de motifs premiers est proposé dans [1]. En pratique ces motifs permettent de couvrir un plus grand nombres d'observations mais au prix d'erreurs de généralisation (sur des données inconnues).

Les approches présentées jusqu'à maintenant reposent sur des algorithmes d'énumération. D'autres approches utilisent des problèmes d'optimisation pour trouver le motif optimisant un certain critère. À chaque fois qu'un motif est généré, des contraintes sont ajoutées pour pourvoir trouver un autre motif. La procédure est répétée jusqu'à avoir suffisamment de motifs.

Bonates *et al.* [10] proposent un programme en nombres entiers pour trouver le motif avec une couverture maximale contenant une observation O donnée en entrée. La procédure de génération est répétée pour toutes les observations positives.

Ryoo *et al.* [81] proposent d'autres modèles de programmes linéaires en nombres entiers pour générer des motifs de couverture maximale. La principale différence entre leurs programmes et celui de Bonates *et al.* est la prise en compte des contraintes utilisateur sur la prévalence, l'homogénéité, le degré.

Hansen *et al.* [43] proposent un algorithme reposant sur une génération de colonnes. Cet algorithme génère un modèle minimisant le nombre d'erreurs tout en étant simple à paramétrer.

2.2.3 Sélection d'un modèle

La Définition 2.5 introduit le concept de modèle comme l'association d'une théorie positive et d'une théorie négative. En pratique nous allons chercher des modèles minimaux (en nombre de motifs) qui permettent d'expliquer toutes les observations. Un grand principe de l'analyse de données est le rasoir d'Occam [92] qui stipule que les hypothèses les plus simples sont les plus vraisemblables. Pour nous, une hypothèse simple est constituée de peu de motifs ayant un degré faible. D'autre part, le point fort de l'ACD consiste en la possibilité d'interpréter simplement ses modèles. Il est clair qu'un modèle constitué de quelques motifs est plus simple à lire et à comprendre qu'un modèle avec plusieurs centaines de motifs.

Dans la section précédente nous avons déjà vu quelques méthodes de génération de motifs qui tiennent compte de la structure du modèle. C'est le cas par exemple de l'algorithme de génération de colonnes de Hansen *et*

al [43]. Dans les autres approches classiques, la génération de motifs est indépendante de la sélection du modèle. Sélectionner un modèle de cardinalité minimale est exactement le problème de couverture minimum d'ensemble. Ce problème déjà évoqué plus tôt est NP-difficile (SP5 [35]).

En pratique, lorsque le nombre de motifs n'est pas trop grand, la sélection d'un modèle se fait par la résolution d'un programme linéaire en nombres entiers [12, 42]. Lorsque le nombre de motifs est trop grand, une solution est d'utiliser une heuristique. Le logiciel Ladoscope indiqué précédemment propose des algorithmes gloutons pour trouver une couverture des observations par les motifs. Dans le Chapitre 3, nous proposons une heuristique utilisant un algorithme de *clustering* pour réduire le nombre de motifs.

2.2.4 Exemple

Nous avons proposé dans l'introduction de ce chapitre un exemple de données issu des Iris de Fisher [33]. Cet extrait ne contenait que 6 observations. Nous allons maintenant considérer toutes ses observations [2] en utilisant les points de coupe proposés dans le Tableau 2.1. Nous ne nous intéressons qu'à séparer deux espèces sur les trois présentes dans la base. L'espèce Setosa est représentée par la classe 1 et l'espèce Versicolor par la classe 0.

Pour éviter les problèmes de sur-apprentissage, nous découpons cinq fois les données en 5-*folds* (voir Chapitre 1). Les motifs sont générés sur chaque *fold* avec une prévalence minimale de 40% et une homogénéité minimale de 80%. En moyenne 200 motifs sont générés.

En considérant l'ensemble des motifs, on obtient un modèle dont la précision est de 95% en entraînement et 95% sur la base de test. Pour avoir un modèle interprétable, nous construisons un modèle minimal en résolvant le problème de couverture minimale où chaque observation doit être couverte par au moins un motif. Le nouveau modèle est constitué d'en moyenne 4.7 motifs avec une précision d'entraînement de 96.6% et 96.9% en généralisation.

À partir de ces modèles, il est possible de construire les règles de classification suivantes en ne conservant que les motifs les plus fréquents parmi les différents *fold* :

- Si la longueur du sépale est supérieure ou égale à 5 et la largeur du sépale inférieure à 3.25 alors l'espèce est Versicolor,
- Si la longueur du sépale est inférieure à 6 et la largeur du sépale supérieure à 3 alors l'espèce est Setosa,
- Dans tous les autres cas, l'observation n'est pas classée.

La précision de ces règles simples est de 94% sur la base complète. Les deux motifs correspondant à ces règles sont présentés sur la Figure 2.1.

2. `http://archive.ics.uci.edu/ml/datasets/Iris`

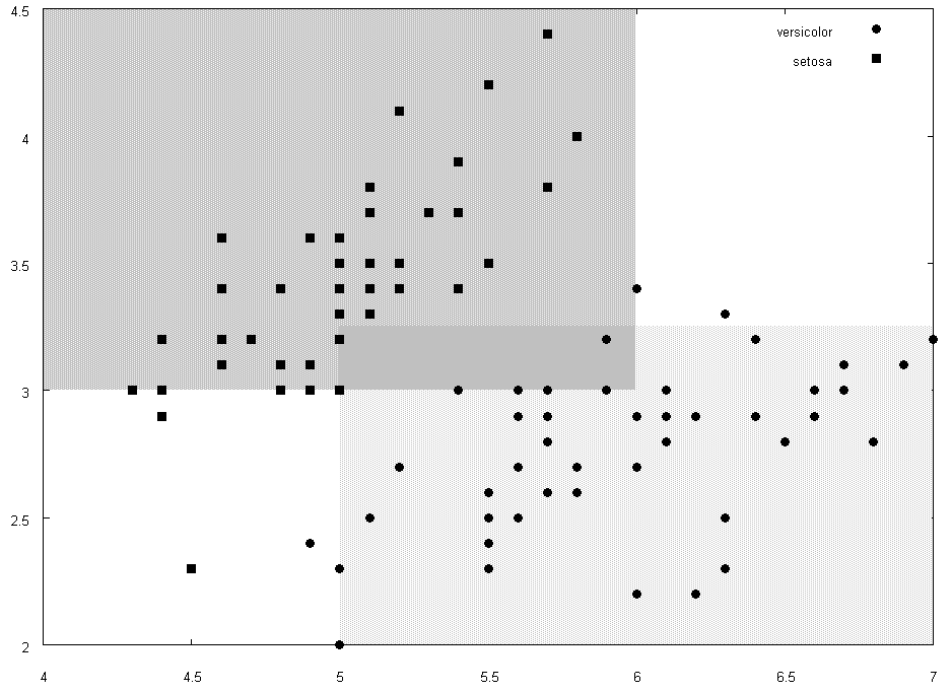

Figure 2.1 – Problème de classification Setosa contre Versicolor. Les deux motifs sont représentés par des rectangles. L'axe des abscisses représente la longueur du sépale, l'axe des ordonnées sa largeur.

2.3 Génération de modèles avec vastes marges

Cette section présente une méthode de génération de modèles de l'ACD à vastes marges, c'est-à-dire avec un fort discriminant. La marge est définie comme la plus petite valeur absolue atteinte par le discriminant sur les données d'apprentissage. La théorie montre que de tels modèles fournissent de faibles erreurs de généralisation [5].

L'idée derrière cette méthode est de générer des motifs avec une forte homogénéité et de ne garder dans le modèle qu'un ensemble de motifs avec peu de conflits (observations couvertes par des motifs positifs et négatifs). Ce travail se concentre essentiellement sur la génération de motifs avec une forte homogénéité et utilise une heuristique gloutonne pour la sélection du modèle. Les algorithmes proposés sont ensuite comparés aux résultats de la littérature.

Nous allons considérer une base de données Ω avec n observations réparties en deux classes Ω^+ et Ω^- telles que $\Omega^+ \cap \Omega^- = \emptyset$. Chacune de ces observations est décrite par un vecteur d'attributs $Z \in \{0,1\}^m$ et une classe $C \in \{0,1\}$. Pour préciser les notations, le vecteur d'attributs Z de

l'observation o sera noté Z_o. La valeur de l'attribut i de l'observation o sera noté $Z_{o,i}$.

Nous proposons un algorithme pour générer des motifs avec une forte homogénéité et une bonne prévalence. L'algorithme de génération de motifs est d'abord présenté sous sa forme théorique, malheureusement inutilisable en pratique. Nous montrons ensuite comment le rendre utilisable en faisant de l'échantillonage aléatoire des observations tout en garantissant avec forte probabilité de ne pas oublier de "bons" motifs. Nous proposons un algorithme de sélection de modèles maximisant les marges du discriminant en minimisant le nombre de conflits entre les motifs du modèle.

L'algorithme exact de génération de motifs présenté dans cette partie a été initialement conçu par Yves Crama et Endre Boros mais n'a jamais été publié. Lors de mon séjour dans le laboratoire RUTCOR, nous avons initié avec Endre Boros les premiers travaux visant à rendre cet algorithme utilisable en pratique. Ce travail est un travail en cours et les propositions énoncées servent à donner l'intuition derrière la méthode retenue.

2.3.1 Algorithme exact pour la génération de motifs

Soient t et f deux observations appartenant respectivement à Ω^+ et Ω^-. Notons $\delta_{t,f}$ l'ensemble des indices des attributs valant 1 pour l'observation t et 0 pour l'observation f. De la même manière, $\rho_{t,f}$ est l'ensemble des indices des attributs valant 0 pour t et 1 pour f. Puisque les ensembles Ω^+ et Ω^- ne s'intersectent pas, il existe alors un attribut i tel que $Z_{t,i} \neq Z_{f,i}$. Par conséquent au moins un des deux ensembles $\delta_{t,f}$ ou $\rho_{t,f}$ n'est pas vide.

Nous définissons la fonction booléenne $\varphi_{t,f} : \{0,1\}^m \rightarrow \{0,1\}$ de la manière suivante :

$$\varphi_{t,f}(x) = \bigwedge_{i \in \delta_{t,f}} x_i \bigwedge_{j \in \rho_{t,f}} \overline{x}_j$$

Par définition, cette fonction prend la valeur 1 lorsqu'elle est évaluée en Z_t et 0 lorsqu'elle est évaluée en Z_f.

Considérons maintenant la fonction booléenne ϕ_t définie comme la disjonction des conjonctions $\varphi_{t,f}$:

$$\phi_t(x) \quad = \quad \bigvee_{f \in \Omega^-} \varphi_{t,f}(x)$$

Soit ϕ_t^d la fonction booléenne duale de ϕ_t définie par : $\phi_t^d(x) = \overline{\phi_t(\overline{x})}$. Cette définition de fonction booléenne duale est classique, elle est équivalente à inverser les signes $\vee$ et $\wedge$ dans l'expression de ϕ_t [25].

Proposition 2.1
La fonction ϕ_t^d est une extension de la fonction booléenne partiellement définie qui associe la valeur 1 à l'observation t et la valeur 0 à toute observation de Ω^-.

Preuve. Par définition, nous avons l'expression suivante de ϕ_t^d :

$$\phi_t^d(x) = \overline{\bigvee_{f \in \Omega^-} \varphi_{t,f}(\overline{x})} = \bigwedge_{f \in \Omega^-} \overline{\varphi_{t,f}(\overline{x})} = \bigwedge_{f \in \Omega^-} \varphi_{t,f}^d(x)$$

Pour toute observation f de Ω^-, $\varphi_{t,f}^d(x) = \bigvee_{i \in \delta_{t,f}} x_i \bigvee_{j \in \rho_{t,f}} \overline{x}_j$, d'où $\varphi_{t,f}^d(Z_t) = 1$ et $\varphi_{t,f}^d(Z_f) = 0$. Puisque ϕ_t^d est une conjonction des disjonctions $\varphi_{t,f}^d$, on a bien $\phi_t^d(Z_t) = 1$ et $\phi_t^d(Z_f) = 0$ pour tout $f \in \Omega^-$. □

Nous allons considérer dans la suite ces fonctions sous leur forme normale disjonctive (disjonction de conjonctions) ou comme ensemble de conjonctions. L'expression de ϕ^d sous forme normale disjonctive que nous allons retenir est celle produite en inversant les signes $\vee$ et $\wedge$ dans l'expression de ϕ et en multipliant les termes. Il s'agit de la procédure SD-Dualization proposée par Crama *et al.* [25] sans absorption. Remarquons que l'expression de ϕ sous forme normale disjonctive est compacte alors que ϕ^d peut avoir un nombre exponentiel de termes. Pour s'en convaincre, il suffit de considérer la fonction $\phi = (x_0 \wedge x_1) \bigvee ... \bigvee (x_{2p} \wedge x_{2p+1})$ dont la fonction duale ϕ^d sous forme normale disjonctive contient 2^{p+1} conjonctions.

On peut déduire de la Proposition 2.1 que toutes les conjonctions apparaissant dans l'expression de ϕ_t^d sous forme normale disjonctive sont des motifs purs positifs couvrants t. En effet puisque $\phi_t^d(Z_f) = 0$, il faut que chaque conjonction de ϕ_t^d prenne la valeur 0 lorsqu'elle est évaluée en Z_f pour $f \in \Omega^-$. D'autre part, les conjonctions de ϕ_t^d sont constituées d'un littéral de chaque disjonction $\varphi_{t,f}^d$ pour $f \in \Omega^-$. Par définition de $\varphi_{t,f}$, chaque conjonction de ϕ_t^d vaut 1 lorsqu'elle est évaluée en Z_t. C'est exactement la définition d'un motif pur couvrant l'observation t.

On peut déduire des propositions précédentes un algorithme de génération de motifs. Cet algorithme calcule pour toute observation t de Ω^+ la fonction booléenne ϕ_t. À partir de cette fonction, il calcule ensuite ϕ_t^d sous forme normale disjonctive, en inversant les signes $\vee$ et $\wedge$ puis en multipliant sans simplification les termes. La liste des conjonctions de ϕ_t^d constitue un ensemble de motifs positifs purs pour Ω. De la même façon il est possible de générer un ensemble de motifs négatifs purs pour Ω.

Cette procédure de génération de motifs est à rapprocher de la méthode proposée dans [24] permettant de générer l'ensemble des motifs purs minimaux couvrant chaque observation. Les motifs générés sont construits à partir de l'ensemble des solutions minimales d'un programme mathématique à variables booléennes défini pour chaque observation.

Proposition 2.2
Soient P un motif généré par l'algorithme précédent à partir de Ω^+ et de Ω^- et $t \in \Omega^+$ une observation couverte par P, alors $P \in \phi_t^d$ (*i.e.* P est une conjonction de ϕ_t^d).

Preuve. Puisque P fait partie des motifs générés par l'algorithme à partir des ensembles Ω^+ et Ω^-, il existe une observation $o \in \Omega^+$ telle que $P \in \phi_o^d$. Si $o = t$ alors le résultat est immédiat.

Notons ψ la conjonction du motif P. Pour chaque littéral de ψ il existe au moins une observation $f \in \Omega^-$ telle que ce littéral apparaisse dans $\varphi_{o,f}^d$ par définition de P. Puisque $\psi(Z_t) = 1$, ce même littéral doit apparaitre dans $\varphi_{t,f}^d$. Ainsi puisque $P \in \phi_o^d$, P est une conjonction de ϕ_t^d. □

Ω	z_0	z_1	z_2	$\mathcal{C}$
O_1	1	1	1	1
O_2	1	0	0	0
O_3	0	0	1	0
O_4	1	1	0	0

Tableau 2.2 – Données fictives avec deux classes et attributs booléens.

À titre d'exemple, nous allons générer les motifs correspondant à un jeu de données simplifié présenté dans le Tableau 2.2. Les conjonctions φ correspondant à l'observation O_1, sont les suivantes :

$$\begin{aligned} \varphi_{1,2}(x) &= x_1 \wedge x_2 \\ \varphi_{1,3}(x) &= x_0 \wedge x_1 \\ \varphi_{1,4}(x) &= x_2 \end{aligned}$$

La fonction ϕ_1 sous forme normale disjonctive est donnée par $\phi_1(x) = x_1 \wedge x_2 \bigvee x_0 \wedge x_1 \bigvee x_2$. La fonction duale sous forme normale disjonctive est donnée par l'expression booléenne :

$$\phi_1^d(x) = x_0 \wedge x_1 \wedge x_2 \bigvee x_0 \wedge x_2 \bigvee x_1 \wedge x_2$$

Les motifs purs couvrant l'observation O_1 sont lus comme les conjonctions de ϕ_1^d : $x_0 \wedge x_1 \wedge x_2$, $x_0 \wedge x_2$ et $x_1 \wedge x_2$.

L'algorithme théorique de génération de motifs proposé repose sur l'énumération des conjonctions de ϕ_t^d pour toutes les observations de Ω^+. De la même manière, il est possible d'énumérer les motifs purs couvrant les observations de Ω^-. Dans le pire des cas, le nombre de conjonctions de ϕ_t^d est exponentiel en la cardinalité de Ω^-. On ne peut donc pas espérer d'algorithme polynomial en la taille de l'entrée pour énumérer tous ces motifs. Nous verrons dans la section suivante comment réduire la complexité de cet algorithme pour le rendre utilisable en pratique.

2.3.2 Considérations pratiques

L'algorithme présenté dans la section précédente permet d'énumérer des motifs purs caractérisant les observations dans la base de données Ω^+. Malheureusement cet algorithme ne peut être employé en pratique puisque son temps d'exécution dépend de manière exponentielle de $|\Omega^-|$. Dans cette section nous étudions de manière théorique le comportement de cet algorithme lorsqu'il est appliqué à un sous-ensemble aléatoire de Ω. L'objectif étant de le rendre utilisable en pratique tout en garantissant la génération de "bons" motifs.

La méthode mise en place consiste à appliquer l'algorithme exact sur plusieurs échantillons aléatoires de Ω. En réduisant le jeu de données, le risque est de générer de nombreux "mauvais" motifs. Ces motifs peuvent être filtrés rapidement en les évaluant sur le jeu de données complet. L'essentiel est de ne pas oublier les "bons" motifs à chaque génération sur un échantillon aléatoire de Ω. Les propriétés suivantes donnent des bornes inférieures sur la taille des échantillons de Ω^+ et Ω^- pour ne pas oublier un "bon" motif qui serait généré par Ω.

Réduction de Ω^-

Nous allons d'abord nous intéresser à la réduction de Ω^-. Pour cela, nous considérons $\tilde{F}$ un échantillon aléatoire simple de Ω^- : chaque observation est choisie aléatoirement avec la même probabilité et les tirages sont indépendants et sans remises. Il est clair que si $\tilde{F}$ contient peu d'éléments, le temps d'exécution de la génération de motifs devient raisonnable. En contrepartie, chaque motif généré avec $\tilde{F}$ contient un sous-ensemble des attributs présents dans un des motifs générés avec Ω^-. On parle alors de **sous-motif**.

À titre d'exemple, considérons la base de données du Tableau 2.2 et $\tilde{F}$ le sous-ensemble de $\{O_2, O_3, O_4\}$ constitué uniquement de l'observation O_2. Les motifs générés à partir de $\tilde{F}$ sont les motifs $P_1 = x_1$ et $P_2 = x_2$. Ces motifs sont clairement des sous-motifs du motif $x_1 \wedge x_2$ généré à partir de Ω^-. D'autre part ces motifs ne sont pas purs si on considère la base de données complète Ω puisque l'observation O_3 est couverte par P_2 et O_4 par P_1.

Soit $L(i) \subseteq \Omega^-$ l'ensemble des observations négatives couvertes par un motif P lorsque le littéral d'indice i est retiré de sa conjonction. Un motif P généré par Ω^- est un **bon motif** de rang l si pour tout attribut i de P, $|L(i)| \geq l$. Nous allons donner des conditions sur $|\tilde{F}|$ pour qu'un bon motif P, généré par Ω^-, soit généré avec une grande probabilité.

Proposition 2.3
Soit P un bon motif de rang l et de degré d généré par Ω. Avec probabilité $1 - \epsilon$, l'union de Ω^+ et d'un échantillon aléatoire simple $\tilde{F} \subseteq \Omega^-$ génère P si :

$$l|\tilde{F}| \geq -\ln(\epsilon/d) \times |\Omega^-|$$

Idée de preuve. Soit E l'événement "P est généré par $\tilde{F}$" et $\bar{E}$ l'événement contraire "P n'est pas généré par $\tilde{F}$". Si P n'est pas généré par $\tilde{F}$ alors il existe au moins un attribut i tel que aucune observation de $L(i)$ ne soit incluse dans $\tilde{F}$. Pour un attribut i de P la probabilité que $L(i)$ ne soit pas inclus dans $\tilde{F}$ est bornée par :

$$p(L(i) \cap \tilde{F} = \emptyset) \leq \left(1 - \frac{|L(i)|}{|\Omega^-|}\right)^{|\tilde{F}|} \leq \left(1 - \frac{l}{|\Omega^-|}\right)^{|\tilde{F}|} \leq e^{-\frac{l|\tilde{F}|}{|\Omega^-|}}$$

En supposant que P soit de degré d, il vient l'inégalité suivante :

$$p(\bar{E}) \leq d \times p(L(i) \cap \tilde{F} = \emptyset) \leq d \times e^{-\frac{l|\tilde{F}|}{|\Omega^-|}}$$

Puisque nous souhaitons vérifier l'inégalité $p(E) \geq 1 - \epsilon$, il suffit de fixer $d \times e^{-\frac{l|\tilde{F}|}{|\Omega^-|}} \leq \epsilon$ et nous obtenons la propriété souhaitée sur $l|\tilde{F}|$.

Ce résultat nous permet d'obtenir à partir d'une probabilité de succès et de propriétés sur les motifs, la taille de l'ensemble $\tilde{F}$ par rapport à la taille de l'ensemble Ω^-. Afin d'augmenter la probabilité de ne pas manquer de bons motifs, une solution consiste à recommencer la génération de motifs à partir d'un nouveau sous-ensemble $\tilde{F}$.

Réduction de Ω^+

Nous allons maintenant voir l'impact sur la génération des motifs positifs de ne travailler qu'avec un sous-ensemble des observations positives. Du point de vue de la complexité, l'expression booléenne duale de ϕ_t doit être calculée pour chaque observation $t \in \Omega^+$. Réduire la taille de Ω^+ permet de réduire le temps d'exécution de l'algorithme de génération de motifs. En pratique cela peut permettre de répéter plusieurs fois les générations de motifs avec de nouveaux sous-ensembles de Ω^+ et de Ω^-.

Considérons P un motif généré par les ensembles Ω^+ et Ω^-. Notons S l'ensemble des observations positives de Ω^+ défini par $S = \{t \in \Omega^+ : P \in \phi_t^d\}$. D'après la Proposition 2.2, S est l'ensemble des observations de Ω^+ couvertes par P. Enfin notons $\tilde{T}$ un échantillon aléatoire simple de l'ensemble Ω^+. La proposition suivante nous donne une borne inférieure sur la taille de l'échantillon pour générer avec forte probabilité le motif P, généré par Ω^+.

Proposition 2.4
Soient P un motif généré par Ω^+ et S l'ensemble des observations de Ω^+

qui génèrent P. Avec probabilité $1-\epsilon$, l'ensemble aléatoire simple $\tilde{T} \subseteq \Omega^+$ permet de générer P si :

$$|\tilde{T}| \geq \frac{-\ln(\epsilon)}{|S|/|\Omega^+|}$$

Idée de preuve. Soit X la variable aléatoire qui compte la cardinalité de $\tilde{T} \cap S$. Par définition de S, le motif P est généré par n'importe quel élément de S et l'ensemble Ω^-. Le motif P est donc généré par $\tilde{T}$ et Ω^- si au moins une observation de S est présente dans $\tilde{T}$, ce qui se traduit par $X > 0$. En notant $\rho = |S|/|\Omega^+|$, la probabilité de l'événement $X = 0$ est bornée par :

$$p(X = 0) \leq (1-\rho)^{|\tilde{T}|} \leq e^{-\rho|\tilde{T}|}$$

Or nous souhaitons choisir la cardinalité de $\tilde{T}$ pour que $p(X > 0) \geq 1-\epsilon$. En résolvant $e^{-\rho|\tilde{T}|} \leq \epsilon$ pour $|\tilde{T}|$, la borne est obtenue.

Remarque. La prévalence de P est égale au rapport $\frac{|S|}{|\Omega^+|}$, puisque l'ensemble S et l'ensemble des observations positives couvertes par P sont égaux.

Ces résultats nous permettent d'avoir la taille d'un sous-ensemble $\tilde{T}$ qui permet avec forte probabilité de générer un motif P. Cette borne dépend uniquement de la prévalence du motif P. En pratique, les motifs intéressants ont une prévalence élevée. Les motifs de prévalence faible caractérisent peu d'observations qui peuvent être simplement des cas isolés.

Utilisation d'un prédicat

Une dernière considération pratique consiste à ne pas générer tous les motifs qui peuvent être générés par les ensembles Ω^+ et Ω^- mais uniquement une partie des sous-motifs vérifiant un prédicat sur le degré maximum, la couverture minimum et la prévalence minimum. Les motifs générés n'ont plus la garantie d'être des motifs purs.

Cette génération de sous-motifs se fait lors de la mise sous forme normale disjonctive de l'expression de ϕ^d. L'algorithme commence avec un motif vide puis ajoute un littéral de chaque disjonction de ϕ^d tant que le prédicat est satisfait et retourne le dernier motif valide lorsque le prédicat devient faux ou lorsqu'un élément de chaque disjonction a été ajouté au motif. Si le prédicat est toujours vrai alors l'ensemble des motifs est généré.

2.3.3 Sélection d'un modèle

L'objectif de la génération de motifs est de fournir de bons motifs. En pratique, le nombre de motifs générés peut devenir grand et des incompatiblités peuvent apparaître à cause de la réduction de Ω^- et de l'utilisation d'un prédicat. Un motif positif et un motif négatif sont dits incompatibles si il existe une observation dans Ω qui appartient à la couverture des deux motifs.

Nous cherchons à sélectionner le modèle contenant un maximum de motifs tout en maintenant le nombre d'incompatibilités inférieur à une valeur seuil dépendante des instances. D'un point de vue théorique, ce problème est une généralisation du problème NP-difficile d'ensemble stable de cardinalité maximale.

Pour résoudre ce problème, nous utilisons une heuristique gloutonne. Les motifs positifs et négatifs sont triés par prévalence décroissante. L'algorithme commence avec un modèle vide, puis ajoute un motif positif et un motif négatif jusqu'à ce que le nombre d'incompatibilités dépasse le seuil donné. À chaque étape, les prévalences sont mises à jour avec les observations non couvertes.

2.3.4 Expérimentations

Dans cette section, nous présentons quelques résultats expérimentaux des méthodes présentées précédemment. Les modèles générés sont évalués en calculant la précision moyenne sur des données classiques de la littérature avec cinq *5-validations croisées*. Les résultats sont comparés aux résultats de Hansen et Meyer [43].

La machine utilisée pour ces expérimentations est un PC avec un processeur Intel Core 2 Duo à 2.4GHz et 4 GO de RAM.

Les données utilisées pour l'expérimentation sont issues de l'*UCI repository*[3]. Les instances sont décrites dans le Tableau 2.3. La première colonne donne l'abréviation utilisée dans les autres tableaux pour nommer les instances, la seconde donne le nom complet de l'instance. Les quatre colonnes suivantes donnent respectivement le nombre d'observations $|\Omega|$, le nombre d'observations de la classe 1 (cardinalité de Ω^+), de la classe 0 (cardinalité de Ω^-) et le nombre d'attributs binaires m. Les données numériques ont été binarisées en utilisant tous les points de coupes à l'aide du logiciel Ladoscope.

Abbréviation	Nom	$\|\Omega\|$	$\|\Omega^+\|$	$\|\Omega^-\|$	m
bcw	Breast Cancer Wisconsin	699	241	458	18
aust	Australian Credit	690	307	383	25
congress	Congressional Voting	435	267	168	16
bupa	Bupa Liver Disease	341	199	142	25

Tableau 2.3 – Description des instances utilisées dans l'expérimentation.

Pour la génération de motifs, 10 sous-ensembles aléatoires simples de 50 observations positives et 10 observations négatives sont utilisés pour chaque instance. Nous utilisons un prédicat sur la prévalence minimum pre_{min} et le degré maximum d_{max}. Les motifs sont ensuite filtrés pour ne garder que

3. `http://archive.ics.uci.edu/ml/`

ceux dont l'homogénéité est au-dessus d'un certain seuil h_{min}. L'ensemble des paramètres de génération de motifs ainsi que le seuil d'incompatibilité pour la sélection du modèle sont récapitulés pour chaque instance dans le Tableau 2.4. Les paramètres de génération de motifs d_{max} et pre_{min} ont été choisis de façon à ce que la génération de motifs se fasse en moins d'une minute. La valeur d_{max} a été fixée à 3 pour garder des motifs simples. La prévalence minimum pre_{min} est fixée initialement à 1 puis diminuée jusqu'à ce que des motifs soient générés. L'homogénéité minimum h_{min} et le nombre de motifs incompatibles sont fixés initialement à 0.75 et 0 et ajustés en fonction de la précision des modèles générés.

Données	d_{max}	pre_{min}	h_{min}	incompatibilité
bcw	3	0.8	0.75	10
aust	3	0.75	0.75	0
congress	3	0.65	0.75	0
bupa	3	0.3	0.5	36

Tableau 2.4 – Paramètres des algorithmes de génération de modèle.

Les performances, en terme de précision, des modèles générés sont présentées dans le Tableau 2.7. Pour chaque instance, les performances sur les données d'apprentissage et de test sont récapitulées. Ces résultats sont les moyennes sur cinq *5-validations croisées*. La dernière colonne contient la précision des modèles de Hansen *et al.* [43] sur les données de test.

Données	Apprentissage	Test	Littérature [43]
bcw	0.948	0.947	0.964
aust	0.855	0.849	0.860
congress	0.949	0.949	0.956
bupa	0.583	0.554	0.716

Tableau 2.5 – Précision des modèles générés sur les données d'apprentissage, de test. La dernière colonne donnant les résultats de la littérature [43].

Les performances des modèles générés sont assez bonnes et proches de la littérature sur trois jeux de données. Sur ces données, les modèles générés sont performants et la différence entre performances d'apprentissage et performances de test est faible. Cela signifie que les modèles générés ne sont pas atteints de sur-apprentissage. La base de données "bupa" est plus difficile que les trois autres. Cette difficulté s'illustre par une baisse de précision pour les résultats de la littérature. Pour notre méthode, il est très difficile de générer de bons motifs sans créer trop d'incompatibilités. Les motifs ainsi obtenus ont des performances faibles par rapport à la littérature.

Cette section a présenté une nouvelle méthode pour l'ACD. Elle repose sur un algorithme exponentiel de génération de motifs. Cet algorithme

est rendu utilisable en pratique, en ne considérant que des sous-ensembles aléatoires des observations. Des premiers résultats théoriques montrent que les motifs générés sont "bons" avec forte probabilité. Les résultats empiriques montrent que cette méthode est performante avec un faible sur-apprentissage mais reste sensible aux données bruitées.

2.4 Application de l'ACD à des données médicales

Cette section présente une application de l'analyse combinatoire de données sur une base médicale. Ce travail a été réalisé en collaboration avec les chercheurs des services de radiologie et de pneumologie de l'hôpital Avicenne pour la détection de pneumopathies interstitielles diffuses (PID) sur des critères radiologiques et cliniques.

2.4.1 Description du problème et des données

La base de données fournie par les médecins contient 124 observations réparties dans 7 classes et une "Divers". Les différentes classes ainsi que la répartition des observations sont données dans le Tableau 2.6. Les observations sont décrites par 79 attributs radiologiques et 27 attributs cliniques. Ces attributs sont tous binaires à l'exception de l'âge qui est un attribut numérique. Cet attribut a été transformé en attributs binaires en utilisant un point de coupe tous les 5 ans.

Classe	Effectif
Sarcoïdose	35
Connectivite	21
Cancer Bronchioloalvéolaire	17
Lymphome	13
COP	11
Pneumopathies à éosinophiles	7
PIDC Médicamenteuses	8
Divers	12

Tableau 2.6 – Répartition des observations dans les 7 classes de la base PID. L'acronyme COP est utilisé pour les Pneumopathies Organisées Cryptogéniques et l'acronyme PIDC pour les Pneumopathies Interstitielles Diffuses Chroniques.

Le problème étant multiclasse, nous avons convenu d'une approche un contre tous. Le problème original est transformé en autant de sous-problèmes que de classes. Chaque sous-problème est un problème de classification binaire qui considère une classe comme positive et les autres comme négatives. Dans ce cas, le sous-problème "Divers" contre tous n'a pas de sens.

Afin de garder des modèles interprétables, nous avons convenu de l'utilisation d'environ quinze motifs par modèle, avec un degré au plus trois. De plus, les motifs présents dans le modèle doivent avoir une sensibilité supérieure à 0.5 et une spécificité supérieure à 0.9 pour garder un sens médical.

L'étude s'est faite en trois grandes étapes pour comparer l'information apportée par les données cliniques et les données radiologiques. Nous avons généré des modèles pour les données cliniques uniquement, pour les données radiologiques uniquement et enfin avec l'ensemble des données.

2.4.2 Génération de motifs par programmation linéaire en nombres entiers

Les modèles ont été générés en adaptant la méthode proposée dans [81]. Dans cette méthode, un motif est généré pour chaque observation qui n'est pas encore couverte en résolvant un programme linéaire en nombres entiers. Cet ensemble de motifs forme le modèle final. Les approches de type *bottom up* et *top down* évoquées dans la Section 2.2.2 ne sont pas envisageables en pratique, à cause du nombre d'attributs et de la condition imposant le degré. Les approches *bottom up* vont générer beaucoup de motifs valides et il ne sera pas possible de sélectionner un modèle compact. Les approches *top down* doivent éliminer un grand nombre d'attributs avant d'obtenir des motifs valides.

L'algorithme de génération de modèle crée, pour chaque observation positive, un motif qui minimise le nombre d'observations positives non couvertes plus le nombres d'observations négatives couvertes à tort. Cette optimisation se fait en respectant les contraintes sur la sensibilité, la spécificité et le degré imposées sur les motifs. Lorsqu'il n'existe pas de motif permettant de couvrir une observation vérifiant les contraintes, l'observation courante est ignorée. Les motifs négatifs sont générés de façon similaire.

Les paragraphes suivants décrivent le programme linéaire en nombres entiers inspiré de [81] permettant la génération de motifs. Afin de simplifier les notations, nous allons considérer uniquement la génération de motifs positifs. Les motifs négatifs se génèrent de façon similaire. Chaque littéral du motif correspondant à l'attribut d'indice i parmi les m attributs de la base est représenté par deux variables de décision booléennes x_i et $\bar{x}_i$. Le littéral x_i sera présent dans le motif si la variable de décision x_i est à 1, son complément sera présent si la variable de décision $\bar{x}_i$ est à 1. Puisque un littéral et son complémentaire ne peuvent être présents en même temps dans un motif, nous ajoutons la contrainte :

$$x_k + \bar{x}_k \quad \leq \quad 1 \quad \forall k \in \{1, ..., m\}. \tag{2.1}$$

Une variable de décision réelle d est ajoutée pour représenter le degré du motif. Ce degré ne doit pas dépasser le degré maximum d_{max}, ce qui nous

amène aux contraintes :

$$\sum_{k=1}^{m} x_k + \bar{x}_k = d \quad \forall k \in \{1, ..., m\}, \tag{2.2}$$

$$d \leq d_{max}. \tag{2.3}$$

Considérons maintenant une observation i dont les attributs sont décrits par le vecteur Z_i. La valeur de l'attribut k de l'observation i est donnée par Z_{ik}. Le littéral x_k du motif généré est satisfait par l'observation i si le produit $x_k Z_{ik}$ vaut un. Pour le littéral $\bar{x}_k$, il faut que le produit $\bar{x}_k(1 - Z_{ik})$ fasse un. Par définition, une observation est couverte par un motif si et seulement si l'ensemble de ses littéraux sont satisfaits par l'observation. Pour chaque observation d'indice i appartenant à Ω^+, nous ajoutons au modèle une variable de décision booléenne y_i qui prend la valeur 1 lorsque l'observation i n'est pas couverte par le motif et 0 sinon. De la même façon, la variable de décision booléenne w_j vaut 1 si l'observation d'indice j de Ω^- est couverte (à tort) par le motif. Ces remarques nous permettent de formuler les deux contraintes suivantes, la première devant être satisfaite par les observations positives, la seconde par les observations négatives :

$$\sum_{k=1}^{m} \{Z_{ik} x_k + (1 - Z_{ik})\bar{x}_k\} + y_i M \geq d \quad i \in \Omega^+, \tag{2.4}$$

$$\sum_{k=1}^{m} \{Z_{jk} x_k + (1 - Z_{jk})\bar{x}_k\} - w_j M \leq d - 1 \quad j \in \Omega^-, \tag{2.5}$$

avec M une "grande" constante. En pratique, il est suffisant de fixer M à la valeur d_{max}.

À ces contraintes, nous ajoutons des contraintes spécifiques à notre application sur la sensibilité minimum Se_{min} et la spécificité minimum Sp_{min}. On peut remarquer que le terme $\sum_{i \in \Omega^+} y_i$ représente le nombre de faux négatifs du motif. De même $\sum_{j \in \Omega^-} w_j$ représente le nombre de faux positifs du motif. En utilisant les définitions de sensibilité et de spécificité, nous obtenons les contraintes suivantes :

$$|\Omega^+| - \sum_{i \in \Omega^+} y_i \geq Se_{min} \times |\Omega^+|, \tag{2.6}$$

$$|\Omega^-| - \sum_{j \in \Omega^-} w_j \geq Sp_{min} \times |\Omega^-|. \tag{2.7}$$

Enfin pour trouver le motif minimisant le nombre d'erreurs (faux négatifs plus faux positifs) couvrant l'observation $o \in \Omega^+$, nous résolvons le programme linéaire en nombres entiers suivant :

$$\min_{x,y,w,d} \left\{ \sum_{i \in \Omega^+} y_i + \sum_{j \in \Omega^-} w_j : (2.1) - (2.7), y_o = 0 \right\}. \tag{2.8}$$

2.4.3 Résultats

Les modèles ont été générés avec l'algorithme décrit dans la Section 2.4.2. Les performances sont évaluées avec la technique du *leave one out* qui correspond à un *n-fold* avec n le nombre d'observation dans la base. Cette méthode a l'avantage de fournir des résultats pertinents lorsque la population d'une des classes est faible.

Dans une première expérience, nous avons comparé les modèles générés sur l'ensemble des données avec les arbres de décision. Les motifs générés doivent respecter un degré maximum de 3, une sensibilité minimum de 0.5 et une spécificité minimum de 0.9. Pour certaines pathologie, il n'existe pas de motif vérifiant ces critères. Ils ont été abaissés à une sensibilité minimum de 0.3 et une spécificité minimum de 0.5 pour le lymphome. Pour les COP, les critères utilisés sont 0.3 pour la sensibilité minimum et 0.8 de spécificité minimum. Les performances des modèles entrainés sur les données clinique et radiologique sont présentées dans le Tableau 2.7. Ces résultats sont comparés à ceux des arbres de décision générés par l'algorithme ID3 de Weka [4]. Les résultats présentés sont les moyennes obtenues en validation croisée (*leave one out*). Les classes étant très déséquilibrées, nous avons ajouté un troisième modèle pour comparer les résultats. Ce modèle, dit "NUL", donne toujours la classe majoritaire.

Pathologie	ACD	ID3	NUL
Sarcoïdose	0.87	0.83	0.72
Connectivite	0.85	0.81	0.83
Cancer Bronchioloalvéolaire	0.90	0.90	0.86
Lymphome	0.86	0.78	0.90
COP	0.88	0.90	0.91
Pneumopathies à éosinophiles	0.94	0.98	0.94
PIDC Médicamenteuses	0.94	0.93	0.93

Tableau 2.7 – Précision des modèles générés par l'ACD, ID3 et le modèle NUL pour chaque pathologie avec les attributs cliniques et radiologiques (validation croisée).

Les performances des motifs générés sont comparables aux arbres de décision. À l'exception des classes lymphome et COP, les modèles sont toujours meilleurs que le modèle NUL. Pour ces deux pathologies, il n'existe pas de motif de degré 3 vérifiant les critères de spécificité et de sensibilité. Une solution serait d'utiliser des motifs de plus grand degré, avec le risque de faire du sur-apprentissage et de perdre en lisibilité.

La seconde expérience consiste à comparer les modèles de l'ACD lorsqu'ils sont entrainés sur les données radiologiques, les données cliniques et

4. `http://www.cs.waikato.ac.nz/ml/weka/`

sur la base complète. Les précisions des modèles générés sont présentées Tableau 2.8. Elles ont été obtenues par validation croisée avec la méthode du *leave one out*.

Les résultats indiquent que pour les pathologies cancer bronchioloalvéolaire et PIDC médicamenteuses, les données radiologiques seules sont plus performantes que les données cliniques seules. Pour les pathologies restantes, les données cliniques seules sont plus performantes. À l'exception des lymphomes et des COP, les modèles générés par les données cliniques et radiologiques sont plus performants que les modèles générés sur un seul type de données. Pour les classes lymphomes et COP, nous avions déjà mis en évidence la difficulté de générer des motifs et modèles pertinents.

Pathologie	Clinique	Radio	Clinique et Radio
Sarcoidose	0.84	0.79	0.87
Connectivite	0.86	0.79	0.85
Cancer Bronchioloalvéolaire	0.86	0.90	0.90
Lymphome	0.89	0.88	0.86
COP	0.89	0.88	0.88
Pneumopathies à éosinophiles	0.93	0.91	0.94
PIDC Médicamenteuses	0.89	0.94	0.94

Tableau 2.8 – Comparaison des précisions des modèles générés à partir des données cliniques et radiologiques.

Les modèles générés pour cette étude sont en général meilleurs que les modèles issus des arbres de décision et apportent de la connaissance. Les contraintes de sensibilité, spécificité et de degré imposées par les médecins permettent de garantir la pertinence et l'interprétabilité de chaque motif. D'autre part, les modèles générés sont constitués d'une quinzaine de motifs (positifs et négatifs) et restent encore interprétables. La seconde expérience montre que l'ajout des données cliniques aux données radiologiques permet d'augmenter la précision en généralisation des modèles.

L'utilisation de programmes linéaires en nombres entiers pour la génération de motifs nous a permis d'obtenir en un temps raisonnable des modèles performants. En effet, le temps de calcul est important puisqu'il faut générer un modèle pour chaque pathologie, en utilisant les données uniquement cliniques, uniquement radiologique et l'ensemble de la base. Pour obtenir des résultats pertinents en validation croisée sur des classes avec des faibles effectifs nous avons choisi la technique du *leave one out* qui demande de calculer un modèle pour chaque observation dans la base.

Les résultats de cette étude vont être présentés aux journées françaises de radiologie et un article médical est en préparation.

2.5 Conclusion

L'analyse combinatoire de données est une méthode d'exploration de données basée sur des règles booléennes. Cette méthode présente l'avantage de fournir des modèles simples, interprétables et performants. Elle est ainsi particulièrement adaptée au diagnostic médical puisqu'elle justifie l'attribution d'une classe à l'aide de motifs. Ils sont alors interprétés comme des arguments en faveur ou en défaveur de l'attribution d'une classe.

Nous avons vu dans la Section 2.2 les différentes méthodes de la littérature pour préparer les données, générer des motifs et sélectionner un modèle. Ces techniques reposent sur des méthodes classiques de la recherche opérationnelle : algorithmes gloutons, programmation linéaire en nombres entiers, etc. La Section 2.3 présente un nouvel algorithme exponentiel de génération de motifs ayant recours à de l'échantillonnage pour le rendre utilisable en pratique. Les premiers résultats sur les données de la littérature sont encourageants. La méthode reste à approfondir, en particulier sur les jeux de données difficiles. La Section 2.4 présente une application de l'ACD sur des données médicales originales. L'algorithme utilisé est un algorithme de la littérature modifié pour répondre aux contraintes imposées par les médecins. Les résultats sont bons et l'étape suivante consiste à interpréter les modèles et éventuellement proposer de nouvelles contraintes.

L'analyse combinatoire de données est une source de problèmes à la fois théoriques comme nous le verrons plus tard, mais aussi pratiques avec la création d'algorithmes de génération de motifs et de sélection de modèle. Il reste encore de nombreuses perspectives, en particulier l'exploitation de modèles à vastes marges en utilisant les résultats théoriques de [5].

Chapitre 3

Analyse de temps de survie

L'analyse de temps de survie[1] est un problème d'exploration de données qui s'intéresse à l'étude du temps précédant un événement. Les applications se situent dans le domaine médical, la finance, la sociologie ou la fiabilité. En médecine on cherche à modéliser le temps avant une rechute, le temps avant un décès suite à une opération ou encore la durée d'une grossesse. En fiabilité, on s'intéresse à modéliser le temps avant la prochaine panne alors qu'en sociologie, on modélise le temps précédant une récidive. Ce problème d'analyse de temps de survie a été très étudié dans la littérature sous l'angle statistique, le lecteur peut se reporter au livre de référence de Klein et Moeschberger [51] pour les techniques classiques ainsi que pour quelques applications.

La prédiction du temps de survie se fait en pratique à l'aide d'un ensemble de données collectées lors d'une étude. Ce problème s'apparente au problème classique de régression présenté dans le Chapitre 1 à l'exception de la présence de données censurées. Une observation censurée est une observation pour laquelle seule une information partielle est connue sur le temps de survie, typiquement une borne inférieure, ce qui correspond au fait que, jusqu'à présent, l'événement dont on souhaite prédire la survenue n'a pas encore eu lieu. En pratique, l'étude permettant de collecter des données ne peut pas durer jusqu'à ce que toutes les observations aient l'événement attendu. De plus, une observation peut quitter l'étude avant que l'événement n'ait eu lieu. On parle dans ce cas de censure à droite, un exemple est présenté Figure 3.1. Dans cette étude fictive, quatre observations sont susceptibles d'avoir l'événement durant la période considérée. À la fin de celle-ci, l'événement est survenu seulement pour les observations 1 et 2. Pour les observations 3 et 4, l'étude a pris fin avant que l'événement n'ait eu lieu. Pour ces observations, nous disposons d'une borne inférieure sur le temps avant l'événement. D'autres formes de censures sont étudiées dans la littérature, la censure à gauche et la censure par intervalles [51].

En écartant les données censurées de l'étude, le problème se réduit à un

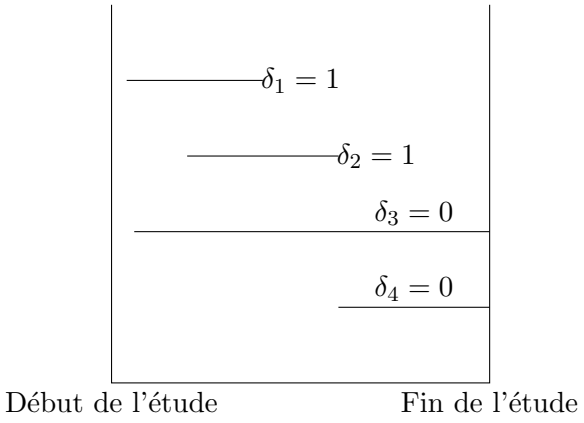

Figure 3.1 – Exemple de censure à droite. L'indicateur δ vaut 1 pour les observations dont l'événement est survenu et 0 pour les observations censurées.

problème de régression. La prise en compte des informations censurées est le principal défi de l'analyse de temps de survie. Ce chapitre présente une extension de l'analyse combinatoire de données pour traiter ce problème d'extraction de données. Elle permet de fournir des modèles simples et facilement interprétables. Une première extension de l'ACD (Analyse Combinatoire de Données) pour l'analyse de temps de survie est proposée dans [56, 57] ; nous listons les principaux défauts de cette méthode et nous proposons des solutions pour y remédier. Dans le second chapitre de [56], le concept de famille de motifs est introduit dans le cadre de problèmes de classification. Nous adaptons dans ce chapitre le concept de famille de motifs aux problèmes de temps de survie. La Section 3.1 donne les notations et quelques définitions classiques. Dans la Section 3.2, les méthodes paramétriques ainsi que les critères d'évaluation de la littérature sont brièvement présentés. Nous introduisons notre adaptation de l'ACD aux problèmes de temps de survie dans la Section 3.3. Cette adaptation est ensuite comparée aux méthodes de la littérature sur des bases de données publiques dans la Section 3.4.

3.1 Notations et définitions

Les variables aléatoires X et C représentent respectivement la durée précédant l'événement et la durée avant que la censure n'intervienne. Ces durée sont mesurées à partir de l'introduction de l'observation dans l'étude (début des segments dans la Figure 3.1). Chaque observation est décrite par un triplet (T, Δ, Z) avec $T = \min(X, C)$, Δ une variable prenant la valeur 1 si l'événement a eu lieu et 0 s'il a été censuré. Le dernier élément Z est le vecteur des attributs de l'observation, il sera considéré comme un

vecteur binaire. On pourra se référer à la Section 2.2.1 pour transformer des attributs quelconques en attributs binaires.

Une base de données est donc un ensemble de N observations de la forme $(t_i, \delta_i, z_i)_{i=1...N}$. Un exemple de données de survie (sans attribut Z) est présenté dans le Tableau 3.1, il correspond aux données de la Figure 3.1.

i	1	2	3	4
T_i	5	4	10	3
δ_i	1	1	0	0

Tableau 3.1 – Exemple de données de survie.

Une première quantité permettant de caractériser les propriétés de survie est la fonction de survie.

Définition 3.1. La fonction de survie $S(t) = P(X > t)$ donne la probabilité de survivre jusqu'à la date t.

La Figure 3.2 présente les tracés de deux fonctions de survie. La courbe en pointillé est une représentation d'une fonction de survie après une opération où la probabilité de survie descend rapidement à cause des risques post-opératoires et se stabilise après la convalescence. La courbe en trait plein représente une fonction de survie après un traitement médical où le risque est repoussé dans le temps.

L'aire sous la courbe de la fonction de survie représente l'espérance de la variable X, soit l'espérance du temps de survie :

$$E(X) = \int_0^\infty S(t)dt$$

En pratique il n'est pas possible de connaître ces quantités, on utilise à la place des estimateurs. On note respectivement $\hat{S}(t)$ et $\hat{E}(X)$ des estimateurs de la fonction de survie et de l'espérance du temps de survie.

Notre objectif est de construire un estimateur paramétrique de la fonction de survie $\hat{S}(t|Z)$ qui attribue une fonction de survie à une nouvelle observation (dont on ne connait pas T) en fonction de ses attributs Z.

3.2 Méthodes de la littérature

L'estimateur de Kaplan-Meier [48], aussi appelé *Product-Limit estimator*, permet d'estimer une fonction de survie à partir d'un ensemble d'observations. On suppose que les événements se produisent aux instants $t_1 < t_2 < ... < t_D$, on note alors d_i le nombre d'observations pour lesquelles l'événement a lieu exactement à la date t_i et Y_i le nombre d'observations dont l'événement a lieu à la date t_i ou plus tard.

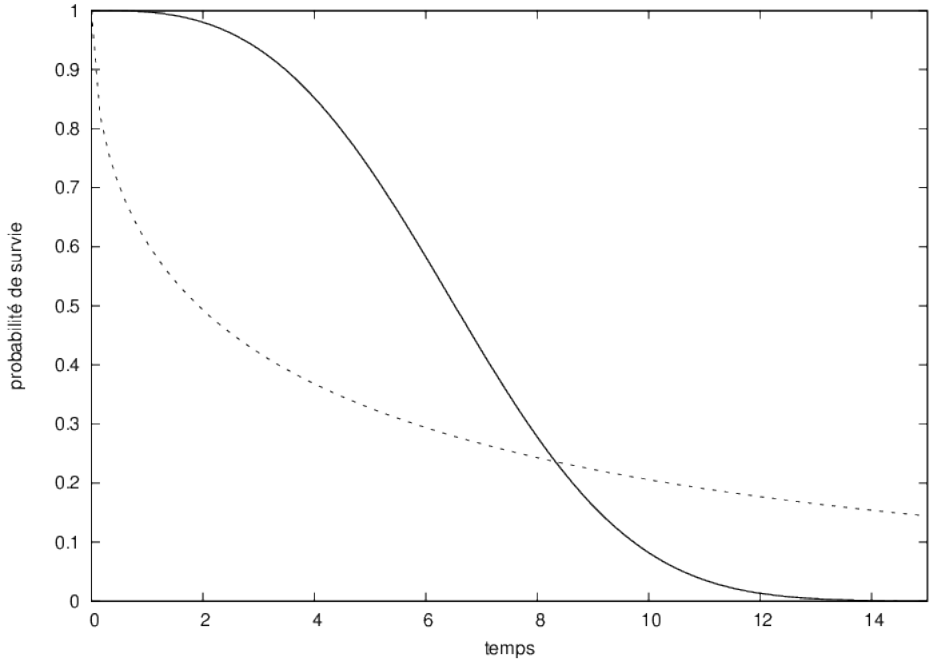

Figure 3.2 – Exemples de fonctions de survie, en trait plein l'effet d'un médicament, en pointillés les suites d'une opération.

Définition 3.2. L'estimateur de Kaplan-Meier $\hat{S}(t)$ est donné par la formule :

$$\hat{S}(t) = \begin{cases} 1 & \text{si } t < t_1 \\ \prod_{t_i \leq t} \left(1 - \frac{d_i}{Y_i}\right) & \text{si } t \geq t_1 \end{cases}$$

Tant qu'aucun événement n'est observé, la fonction de survie est estimée par 1. Le terme $1 - \frac{d_i}{Y_i}$ représente la probabilité conditionnelle de ne pas avoir l'événement à la date t_i sachant que l'événement n'a pas eu lieu avant t_i. La probabilité d'avoir l'événement après la date t est la même que la probabilité de ne pas avoir l'événement avant la date t, qui est estimée par le produit des $1 - \frac{d_i}{Y_i}$. En prenant l'exemple du Tableau 3.2, l'estimateur de Kaplan-Meier donne la fonction de survie tracée sur la Figure 3.3. La probabilité de survivre pendant 15 unités de temps est alors estimée à 0.7.

3.2.1 Méthodes paramétriques

Contrairement à la méthode de Kaplan-Meier qui ne prend pas en compte la valeur des attributs Z, les estimateurs présentés dans cette section utilisent cette information. On parle alors d'estimateurs paramétriques.

La méthode de Cox [51] est un estimateur classique pour modéliser l'influence des attributs sur les propriétés de survie. Il s'agit essentiellement

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T_i	6	6	6	6	7	9	10	10	11	13	16	17	19	20	22	23
δ_i	1	1	1	0	1	0	1	0	0	1	1	0	0	0	1	1

17	18	19	20	21
25	32	32	34	35
1	0	0	0	1

Tableau 3.2 – Données de survie fictives pour 21 observations, sans attribut Z.

d'estimer les coefficients d'un modèle par des méthodes de type maximum de vraisemblance.

Des méthodes plus récentes, basées sur les arbres de décision et les arbres de régression, ont été développées pour l'analyse de temps de survie [58]. Les arbres de décision ont l'avantage de fournir des modèles simples à lire et à interpréter. Les performances des arbres de décision peuvent être renforcées par des techniques de *boosting* consistant à entraîner en parallèle plusieurs arbres. On parle alors de forêts de survie [46].

D'autres méthodes plus complexes comme les réseaux de neurones [80] et les méthodes de Bayes naïves [95] ont été adaptées aux problèmes de temps de survie. Ces modèles sont généralement moins faciles à interpréter.

Une première extension de l'Analyse Combinatoire de Données pour l'analyse de temps de survie (ACDS) a été proposée dans [57]. L'objectif était de développer une méthode performante tout en fournissant un modèle facilement interprétable. La méthode repose sur les deux étapes classiques de l'ACD : une génération de motifs puis une construction du modèle. Les résultats expérimentaux de cette méthode sont comparables aux autres méthodes de la littérature. Son principal défaut concerne sa génération de motifs, basée sur une heuristique gloutonne. L'exploration d'un nombre réduit de motifs peut entraîner des modèles plus complexes, avec des motifs de fort degré.

Pour plus de détails sur cette méthode, le lecteur pourra se référer au chapitre 3 de [56]. Nous proposons dans la suite une autre extension de l'analyse combinatoire de données pour l'analyse de temps de survie.

3.2.2 Mesures de performance

Pour évaluer la performance des estimateurs paramétriques, nous avons à notre dispositions deux critères de la littérature : l'indice de concordance [44] et le *Brier score* [38]. L'indice de concordance mesure la capacité de l'estimateur à restituer l'ordre des prédictions, il ne s'intéresse pas à la qualité de la prédiction. Le *Brier score* quand à lui est une mesure d'inexactitude de la fonction de survie estimée. L'utilisation d'un critère plutôt qu'un autre est

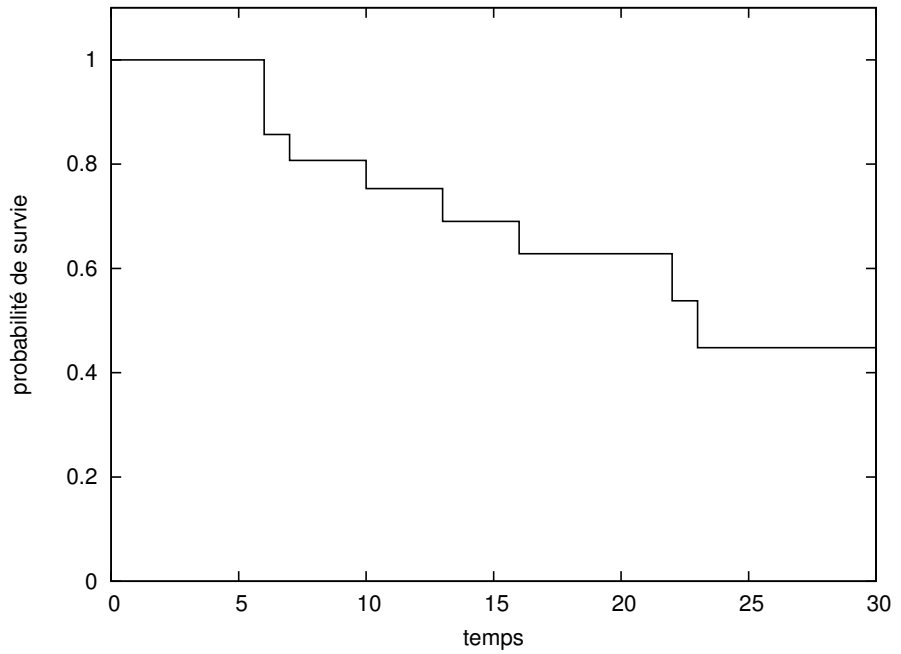

Figure 3.3 – Tracé de l'estimateur de Kaplan-Meier pour les données du Tableau 3.2.

sujet à discussions dans la littérature, dans la suite de ce travail nous avons utilisé l'indice de concordance dont nous donnons la définition formelle.

Définition 3.3. Une paire d'observations (i, j) est **ordonnée** si $T_i < T_j$ et $\delta_i = 1$.

Une observation censurée ne peut être le premier élément d'une paire ordonnée puisque nous ne connaissons pas le temps avant son événement. On note n_o le nombre de paires d'observations ordonnées.

Définition 3.4. Une paire d'observations (i, j) est **concordante** (ou **semi-concordante**) si c'est une paire ordonnée et si $\hat{E}(X|Z_i) < \hat{E}(X|Z_j)$ (ou $\hat{E}(X|Z_i) = \hat{E}(X|Z_j)$) pour un estimateur donné $\hat{E}$.

On note n_c et n_s le nombre de paires d'observations concordantes et semi-concordantes.

Définition 3.5. L'indice de concordance **c-index** est donné par le ratio :

$$\frac{n_c + 0.5\ n_s}{n_o}$$

Les valeurs que peut prendre l'indice de concordance varient entre 0 et 1. Un score de 1 signifie une restitution de l'ordre sans erreur, un score de 0.5 signifie que l'estimateur n'est pas plus performant qu'un ordre aléatoire. Un score entre 0 et 0.5 correspond à un ordre plus mauvais qu'un ordre aléatoire, proche de l'ordre inverse (score de 0).

3.3 Méthode des familles de motifs

Dans cette section, nous proposons une extension de l'ACD pour traiter des problèmes de temps de survie. Cette méthode s'inscrit dans le cadre des méthodes paramétriques. Dans un premier temps il est nécessaire d'adapter les définitions de l'ACD dans le cadre du temps de survie. Nous allons ensuite voir comment générer des motifs de survie tout en contrôlant leur complexité. Une phase d'agrégation en familles permet de grouper des motifs similaires et d'identifier des observations avec des propriétés de survie homogènes. À partir de cette description, il nous est possible d'extraire un modèle court dont la performance est évaluée en utilisant le critère du c-index.

3.3.1 Motifs de survie

Les définitions classiques de l'analyse combinatoire de données ne s'étendent pas simplement à des problèmes de régression ou d'analyse de survie, c'est pourquoi nous distinguons les motifs de survie des motifs classiques. Un motif classique doit couvrir majoritairement des observations de la même classe. La notion de classe n'existant pas dans ce contexte, les motifs de survies devront couvrir des observations avec des caractéristiques de survies similaires. Nous utilisons la définition de motif de survie proposée dans [57], m représente le nombre d'attributs dans la base de données.

Définition 3.6. Un **motif de survie** est une conjonction élémentaire sur les attributs caractérisant des observations avec des temps de survies similaires.

Les définitions de degré et de couverture données dans le Chapitre 2 sont toujours valides dans le cadre de l'analyse de temps de survie. L'homogénéité et la prévalence ne peuvent pas être définies dans ce cadre du fait de l'absence de classes. Nous allons les remplacer par deux nouveaux critères.

Définition 3.7. La **couverture** d'un motif de survie est la proportion d'observations couvertes par rapport au nombre d'observations dans la base de données.

Suivant le contexte, la couverture peut aussi représenter les observations couvertes par le motifs.

Le deuxième paramètre que nous allons utiliser permet de caractériser l'homogénéité des observations décrites par un motif de survie. Pour cela nous allons attribuer à chaque motif la fonction de survie fournie par l'estimateur de Kaplan-Meier en utilisant les observations couvertes par le motif. Nous faisons l'hypothèse que des observations hétérogènes ont une fonction de survie proche de la fonction de survie moyenne alors que des observations homogènes pertinentes (qu'elles soient à fort ou à faible risque) auront une fonction de survie éloignée de la moyenne.

Définition 3.8. La **fonction de survie moyenne** (*baseline*) est la fonction de survie de l'ensemble des observations.

Afin de comparer la proximité de la fonction de survie d'un motif avec la fonction de survie moyenne, nous utilisons le test du logrank [51]. Ce test compare pour deux populations le nombre d'événements observés à chaque instant par rapport au nombre d'événements attendus si les deux populations avaient les mêmes propriétés de survie. En fixant un seuil de confiance (généralement 95%), on obtient une valeur du logrank au-delà de laquelle les deux populations vont être considérées comme significativement différentes (au sens statistique).

Définition 3.9. Le **logrank** d'un motif de survie est la valeur du test du logrank entre les observations couvertes par le motif et l'ensemble des observations.

Une valeur de logrank proche de 0 signifie que le motif couvre des observations proches de la moyenne tandis qu'une valeur plus grande caractérise un motif couvrant des observations significativement différentes (qu'elles soient à fort ou faible risque).

Définition 3.10. Un motif est à **fort risque** (resp. **faible risque**) si sa fonction de survie est en dessous (resp. en dessus) de la fonction de survie moyenne.

Notre objectif avec cette méthode est de parcourir un grand ensemble de motifs. Pour cela nous générons de manière exhaustive tous les motifs jusqu'à un degré fixé d. Un des objectifs de l'analyse combinatoire de données est de fournir des motifs facilement interprétables par l'utilisateur ; en pratique le degré est donc limité à trois ou quatre. Il est possible de réduire le nombre de motifs générés en ignorant ceux dont la couverture est inférieure à une couverture minimum. Enfin il est possible de filtrer les motifs générés en utilisant une valeur de logrank minimum afin d'éliminer les motifs dont les propriétés de survie sont proches de la moyenne.

Pour illustrer les étapes de la méthode, nous considérons un jeu de données fictif sur lequel nous avons généré un ensemble de 11 motifs. Les fonctions de survie de ces 11 motifs sont représentées sur la Figure 3.4. On distingue trois comportement dans ces motifs : quatre courbes à faible risque (Motifs 0 à 3), cinq courbes à risque fort (Motifs 4 à 8) et deux courbes de risque moyen (Motifs 9 et 10) coupant la moyenne en gras.

3.3.2 Création des familles

L'étape de génération de motifs fournit un grand ensemble de motifs dont certains peuvent être redondants : ils caractérisent les mêmes observations mais en utilisant des attributs différents. Dans cette seconde étape,

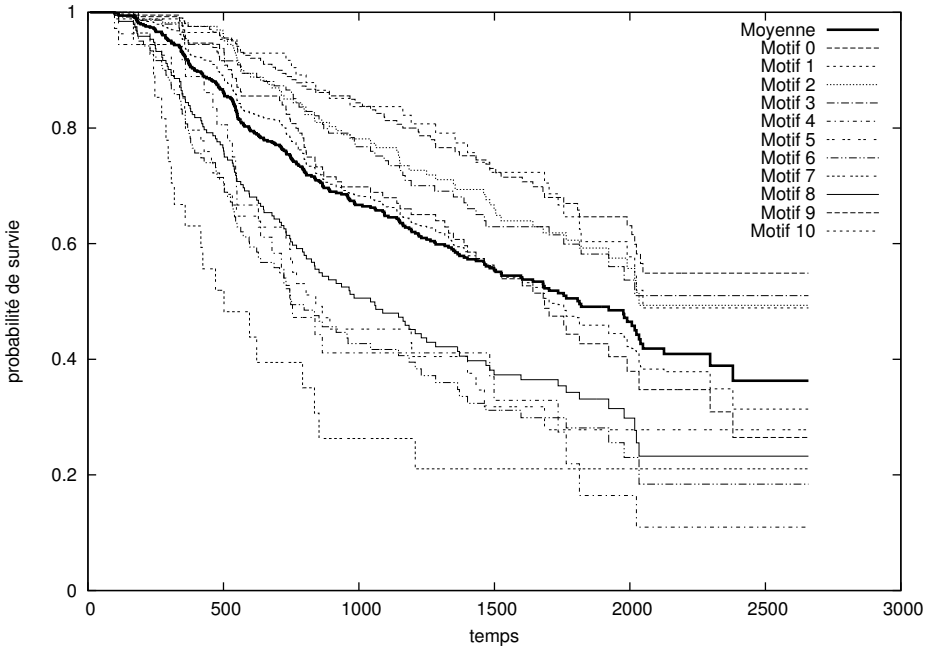

Figure 3.4 – Estimateur de Kaplan-Meier sur les observations couvertes par un motif. En gras la fonction de survie moyenne, les autres courbes représentent les fonctions de survie de chaque motif.

nous allons regrouper ces motifs en famille en utilisant un algorithme de détection de communautés. La détection de communautés est détaillée dans le Chapitre 4.

Nous définissons une mesure de similarité entre deux motifs. Cette mesure est un nombre compris entre 0 et 1, elle est proche de 0 lorsque les motifs sont très différents, et elle s'approche de 1 lorsque les motifs sont similaires. Différents choix de mesure de similarité ont été étudiés dans [27]. Nous avons retenu le coefficient de Jaccard [47] pour sa simplicité d'interprétation. Le terme $cov(P)$ représente la couverture en terme d'ensemble du motif P.

Définition 3.11. L'indice de Jaccard entre deux motifs P_i et P_j est défini par :

$$\mathcal{J}_{ij} = \frac{|cov(P_i) \cap cov(P_j)|}{|cov(P_i) \cup cov(P_j)|}$$

Deux motifs sont synonymes si leur mesure de similarité est supérieure à un seuil t.

La similarité entre les motifs permet de définir un graphe $G = (V, E)$ où chaque sommet de V correspond à un motif P et une arête est incidente à deux sommets représentant des motifs synonymes. Pour continuer l'exemple des motifs de la Figure 3.4, le graphe des similarités est représenté Figure 3.5 avec un seuil de 90%.

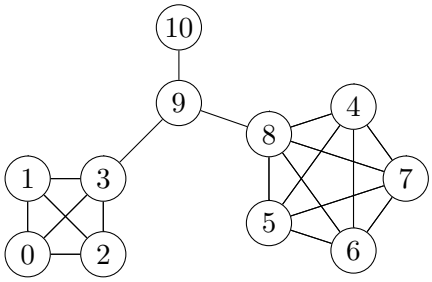

Figure 3.5 – Graphe des similarités.

Une clique du graphe G représente un ensemble de motifs qui sont tous synonymes deux à deux. On pourrait alors définir les familles de motifs comme les classes d'une partition en cliques du graphe G. Cette définition pose deux problèmes : partitionner un graphe en un nombre minimum de cliques est un problème NP-difficile et difficile à résoudre de façon exacte sur plus d'une vingtaine de sommets. Le second problème concerne la définition de clique qui dans notre cas est trop forte, une "presque" clique où quelques arêtes sont absentes pourrait être suffisante.

Une famille de motifs peut également être définie comme un ensemble de sommets fortement connectés, avec peu de connections vers l'extérieur. Pour partitionner le graphe G en familles de motifs, nous utilisons l'algorithme glouton proposé par Newman [69]. L'algorithme commence avec un sommet par famille, puis fusionne les deux familles qui entraînent la plus grande augmentation d'une fonction d'évaluation appelée modularité [71] (notée Q). L'algorithme s'arrête lorsqu'il atteint un maximum local, c'est-à-dire lorsque la fonction d'évaluation ne peut plus augmenter. Cet algorithme a l'avantage d'être rapide, de donner de bons résultats empiriques et peut être appliqué à de grands graphes puisque son temps d'exécution est en $O((m+n)n)$ avec n le nombre de sommets et m le nombre d'arêtes. Il s'inscrit dans le cadre des algorithmes hiérarchiques présentés dans le Chapitre 4 traitant de la détection de communautés.

Pour continuer l'exemple précédent, les différents regroupements effectués par l'algorithme de Newman sont présentés Figure 3.6. On retrouve en abscisse les sommets de G, en ordonnée les valeurs de la modularité Q à chaque fusion opérée par l'algorithme. L'algorithme s'arrête en trouvant trois familles : $\{0,1,2,3\}, \{9,10\}, \{4,5,6,7,8\}$. On remarque que ces familles correspondent à des observations à fort risque $\{4,5,6,7,8\}$, à faible risque $\{0,1,2,3\}$ et à risque moyen $\{9,10\}$.

3.3.3 Sélection d'un modèle

L'étape précédente permet de regrouper des motifs similaires en familles. Malgré cette réduction, le nombre de familles reste en pratique toujours

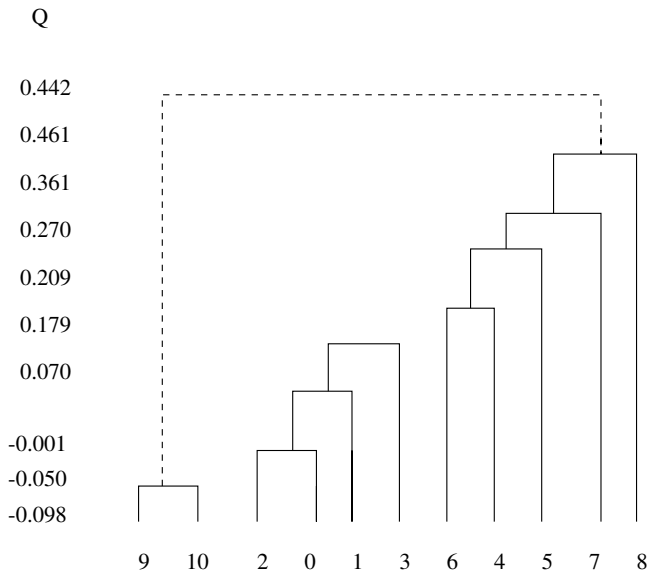

Figure 3.6 – Regroupements successifs lors de l'application de l'algorithme de Newman.

trop grand pour être facilement interprétable. En s'inspirant des méthodes classiques présentées dans le Chapitre 2, nous proposons un modèle de programmation linéaire en nombres entiers permettant la sélection d'un sous-ensemble minimum de familles expliquant les observations à fort risque et à faible risque. Enfin un motif de chaque famille est désigné comme représentant pour être dans le modèle final. La Section 3.4 compare plusieurs méthodes de sélection d'un représentant basées sur le degré, la couverture et le logrank des motifs.

Pour chaque motif m, on note T_m l'aire sous sa courbe de survie (l'espérance du temps avant l'événement). On note T_{HR} le maximum de T_m sur l'ensemble des motifs à fort risque et T_{LR} le minimum de T_m sur l'ensemble des motifs à faible risque.

Définition 3.12. Une observation d'indice i est à **fort risque** si $t_i \leq T_{HR}$ et $\delta_i = 1$, à **faible risque** si $t_i \geq T_{LR}$. Elle est **intéressante** si elle est à fort risque ou à faible risque.

En pratique nous cherchons à couvrir les observations intéressantes à l'aide d'une ou plusieurs familles.

Définition 3.13. Une observation est **couverte** par une famille si elle est couverte par chacun de ses motifs.

La sélection d'un sous-ensemble minimal de familles peut alors être modélisée par un problème de couverture. Pour simplifier les notations, HR

représente les indices des observations à fort risque, LR les indices des observations à faible risque.

Soient N le nombre d'observations, F le nombre de familles et B la matrice de N lignes et F colonnes telle que :

$$(B)_{ij} = \begin{cases} 1 & \text{si la famille } j \text{ couvre l'observation } i \\ 0 & \text{sinon} \end{cases}$$

Le problème de couverture est défini par :

$$\min \sum_{i=1}^{F} x_i$$
$$\text{s.c.} \begin{cases} B_i x \geq 1 & \forall i \in HR \cup BR \\ x \in \{0,1\}^F \end{cases}$$

Chaque famille f est représentée dans le problème de couverture par une variable de décision booléenne x_f. La variable x_f vaut 1 si la famille f est sélectionnée, 0 sinon. La contrainte $B_i x \geq 1$ impose que les observations intéressantes soient couvertes par au moins une famille. Enfin l'objectif de ce problème d'optimisation est de minimiser le nombre de familles sélectionnées.

Le problème de couverture minimum est un problème NP-difficile (voir problème SP5 dans [35]), mais pour le nombre de familles générées, il est possible de le résoudre en un temps raisonnable par un algorithme d'énumération de type *branch & bound*. En pratique un tel problème peut ne pas avoir de solution si une observation ne peut être couverte par aucune famille, nous verrons une formulation plus robuste dans la partie expérimentale (Section 3.4).

Pour obtenir un modèle au sens de l'analyse combinatoire de données, il reste à trouver un motif représentant chaque famille. Ce problème est étudié dans le cadre des problèmes de classification dans le chapitre 2 de [56]. D'après nos premières expériences [27], différentes méthodes de choix du représentant donnent des modèles équivalents. Dans la suite nous avons utilisé le motif qui a le logrank le plus élevé, en pratique on pourrait choisir les représentants de manière à minimiser le nombre d'attributs présents dans le modèle final ou minimiser le degré des motifs.

L'attribution d'une fonction de survie à une nouvelle observation $O = (t, \delta, z)$ se fait en utilisant la formule proposée dans [57]. En notant $\hat{S}(t)$ la fonction de survie moyenne, $\hat{S}_P(t)$ la fonction de survie du motif P et $\mathcal{P}$ l'ensemble des motifs couvrant l'observation O, la fonction de survie attribuée à O est donnée par :

$$\hat{S}(t|z) = \frac{\hat{S}(t) + \sum_{P \in \mathcal{P}} \hat{S}_P(t)}{|\mathcal{P}| + 1}$$

On peut remarquer que la prédiction par défaut est la fonction de survie moyenne qui est ensuite corrigée par les fonctions de survie des motifs couvrant l'observation.

3.4 Expérimentations

Dans cette section, nous comparons les performances de la méthode des familles de motifs de survie proposée en Section 3.3 aux méthodes de la littérature. Nous avons retenu les méthodes à base de règles : les arbres de survie [58], les forêts de survie [46] et l'ACDS (Analyse Combinatoire de Données pour l'analyse de temps de Survie) proposée dans [57]. Les forêts de survie sont calculées en utilisant le logiciel R [79] et le module fourni avec [46]. Le paramétrage de l'algorithme est le même que dans [46] puisque nous utilisons les mêmes bases de données. L'arbre de survie est calculé en utilisant l'algorithme des forêts et en réglant le nombre d'arbres à 1. L'ACDS a été utilisée dans les mêmes conditions expérimentales que dans [57].

L'ensemble des méthodes est testée en utilisant cinq 5-validations croisées (voir Chapitre 1). Les performances sont mesurées sur la base de test en utilisant le c-index.

3.4.1 Environnement de test

La machine utilisée lors de ces expérimentations est une station SUN sous Debian avec un processeur Intel Xeon à 2.43GHz et 4 GO de RAM.

La génération des motifs est effectuée à l'aide du logiciel Ladoscope [1]. Les programmes linéaires en nombres entiers sont résolus avec le logiciel IBM ILOG CPLEX [2].

3.4.2 Bases de données

La base de données GBSG-2 (*German Breast Cancer Study Group 2* [3]) a été initialement étudiée dans [82]. Elle contient 686 observations possédant chacune 8 attributs : 2 booléens, 5 numériques et 1 attribut de catégorie. Sur les 686 observations, 299 ont eu l'événement (une rechute), les autres sont censurées.

La base de données PBC (*Primary Biliary Cirrhosis* [4]) a été utilisée pour comparer l'effet d'un traitement à celui d'un placebo [34]. Elle contient 312 observations dont 125 sont décédées durant l'étude. Pour chaque observation, nous disposons de 14 attributs.

Les données n'étant pas binaires, nous les transformons en utilisant les méthodes classiques décrites dans le Chapitre 2. Pour les données numériques, nous utilisons tous les points de coupe.

1. `http://pit.kamick.free.fr/lemaire/software-lad.html`
2. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`
3. `http://www.blackwellpublishing.com/rss/Volumes/A162p1.htm`
4. `http://lib.stat.cmu.edu/S/Harrell/data/descriptions/pbc.html`

3.4.3 Génération de motifs

La génération des motifs est effectuée sur les deux bases de données avec une couverture minimum de 5% et un degré maximum de 4. L'ensemble des motifs est généré avec le logiciel Ladoscope, 110 794 motifs sont générés pour la base GBSG-2, et 161 693 pour la base PBC. Ce grand nombre de motifs empêche le calcul de la mesure de similarité en un temps raisonnable. Puisque nous cherchons uniquement les motifs caractérisant les populations à risque, nous allons réduire le nombre de motifs en utilisant la valeur du logrank. L'objectif étant de réduire le nombre de motifs à environ 6000, taille au-delà de laquelle le calcul des similarités devient difficile dans notre environnement de test.

La répartition des valeurs du logrank varient en fonction de la couverture des motifs. Nous partitionnons l'espace des motifs en classes de même cardinalité, dans chaque classe nous allons ensuite conserver les motifs avec le plus grand logrank. Il n'est pas vraiment possible de décider combien de motifs conserver dans chaque classe *a priori*. Au final nous retenons 2800 motifs en dessous et 2800 motifs en dessus de la fonction de survie moyenne pour chaque base. La répartition finale est donnée dans le Tableau 3.3, elle est identique pour les motifs à fort risque et à faible risque.

Couverture	GBSG-2	PBC
5% - 6.5%	800	500
6.5% - 8%	800	500
8% - 10%	400	500
10% - 15%	300	500
15% - 20%	300	500
20% - 30%	200	300

Tableau 3.3 – Répartition du nombre de motifs dans chaque classe de couverture.

3.4.4 Détection des communautés dans le graphe des similarités

La construction du graphe des similarité demande de fixer un seuil. Lorsque la mesure de similarité entre deux motifs est au dessus de ce seuil alors une arête est présente dans le graphe. Le nombre d'arêtes en fonction du seuil est donné Figure 3.7 pour le problème GBSG-2. On peut remarquer que dès que le seuil dépasse 0.5, le graphe devient creux (moins de 10% des arêtes par rapport au nombre d'arêtes possibles).

Le nombre d'arêtes du graphe va influer sur le nombre de communautés détectées. Lorsque le graphe est très creux (c'est-à-dire lorsqu'il comporte peu d'arêtes), un grand nombre de petites communautés vont être détectées.

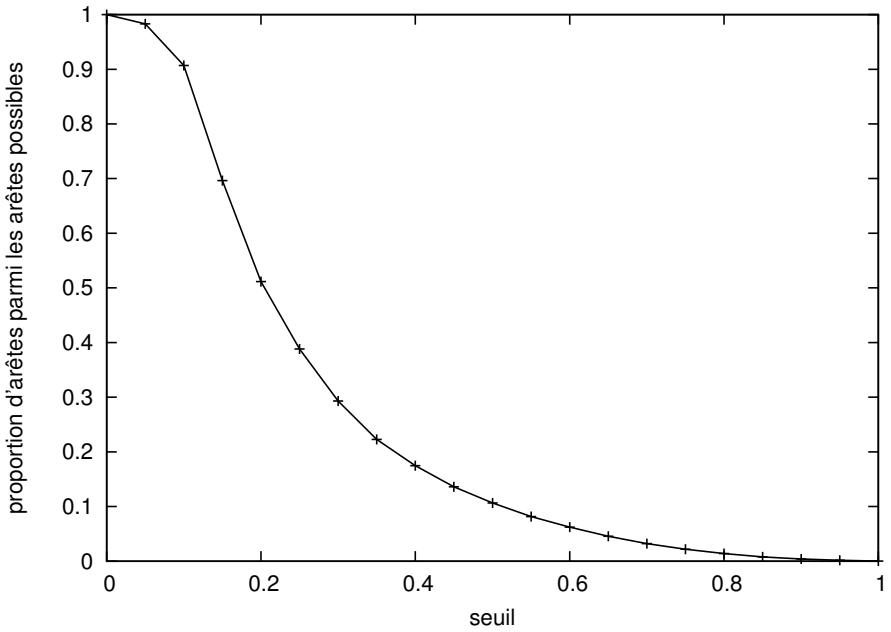

Figure 3.7 – Proportion du nombre d'arêtes de G parmi les arêtes possibles en fonction du seuil (GBSG-2).

À l'inverse lorsque le graphe présente beaucoup d'arêtes, l'algorithme va trouver quelques grandes communautés. L'influence de ce paramètre sur le nombre de communautés et leur taille moyenne est récapitulée dans le Tableau 3.4 pour la base GBSG-2. En choisissant un seuil à 0.85, on réduit le nombre de communautés détectées à 172, avec une taille moyenne assez faible pour être interprété.

Seuil	Nombre de communautés	Taille moyenne des communautés
0.95	406	3.57
0.90	272	7.43
0.85	172	13.06
0.80	109	21.43
0.75	61	38.72
0.70	40	69.53
0.65	25	95.36
0.60	20	119.5

Tableau 3.4 – Influence du seuil sur la partition en communautés (GBSG-2).

Enfin en faisant varier le paramètre autour de la valeur 0.85, il est possible de trouver rapidement le seuil qui donne le modèle avec les meilleures performances (en terme de c-index). Pour la base GBSG-2 le seuil est fixé à

0.85 et pour la base PBC à 0.86.

Il est possible de générer un modèle en sélectionnant un motif dans chaque communauté afin de comparer la partition que nous obtenons avec l'algorithme de Newman et une partition aléatoire. Le motif choisi est celui qui a la plus grande valeur de logrank. Pour ne pas tomber sur un cas particulier les performances du modèle aléatoire sont les performances moyennes sur 20 essais. Les performances des deux modèles sur la base GBSG-2 sont résumées dans le Tableau 3.5. Ces performances sont évaluées en validation croisée sur cinq 5-validations croisées. Le modèle issu des communautés est significativement plus performant que la moyenne des modèles aléatoires.

	Modèle aléatoire	Modèle des communautés
Nombre de motifs	143.3 ± 0.8	179 ± 17
c-index (apprentissage)	0.5273 ± 0.0001	0.7193 ± 0.0063
c-index (test)	0.5408 ± 0.0003	0.6875 ± 0.0325

Tableau 3.5 – Performances des modèles construits à partir d'une partition aléatoire et de la partition en communautés (GBSG-2).

3.4.5 Sélection du modèle

La partition du graphe en communautés permet de réduire l'ensemble des motifs à environ 180 familles. Le modèle est performant comparé à un modèle aléatoire mais il est encore trop grand pour être interprétable. La résolution du problème de couverture se fait avec IBM CPLEX [5].

En pratique, la résolution du problème de couverture présenté dans la section précédente peut devenir plus délicat à cause d'erreurs dans les données. Il est possible par exemple qu'une observation ne puisse être couverte par aucune des familles, dans ce cas le problème n'a pas de solution. On ne cherche donc à couvrir que les observations intéressantes couvertes par au moins γ familles. D'autre part on peut demander à ce que chaque observation soit couverte par un minimum de $\alpha \leq \gamma$ familles, essentiellement pour introduire de la robustesse dans le modèle final.

L'influence des deux paramètres sur le modèle final est présentée Tableau 3.6. On peut remarquer au final que ces deux paramètres n'ont pas d'influence significative sur la qualité en prédiction du modèle aussi bien sur la base d'entraînement que sur la base de test. Le seul effet est sur la taille du modèle final. On retiendra alors les paramètres $\alpha = 1$ et $\gamma = 2$ qui permettent d'obtenir le modèle de plus petite taille sans pour autant sacrifier les performances.

Nous venons de voir comment sélectionner un sous-ensemble minimal de famille de motifs pour construire un modèle ; il faut trouver un motif

5. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`

α	γ	Entraînement	Test	Taille
1	1	0.7079 ± 0.008	0.6849 ± 0.028	17.56 ± 2.47
1	2	0.7003 ± 0.008	0.6832 ± 0.029	11.11 ± 2.00
2	2	0.7111 ± 0.007	0.6864 ± 0.031	26.92 ± 3.53

Tableau 3.6 – Influence des paramètres du problème de couverture sur la performance (c-index) du modèle (GBSG-2).

représentant sa famille. On peut imaginer plusieurs politiques de choix : le plus faible degré, la plus petite couverture, la plus grande couverture, le logrank minimum, le logrank maximum ou bien un choix aléatoire dans la famille. L'influence de ces différentes politiques est présentée Tableau 3.7.

Mode de sélection	Entraînement	Test
Degré min.	0.6987 ± 0.007	0.6800 ± 0.030
Couverture min.	0.6988 ± 0.005	0.6798 ± 0.031
Couverture max.	0.6984 ± 0.008	0.6805 ± 0.031
Logrank min.	0.6965 ± 0.008	0.6760 ± 0.035
Logrank max.	0.7003 ± 0.008	0.6832 ± 0.029
Aléatoire	0.6982 ± 0.010	0.6826 ± 0.030

Tableau 3.7 – Performances (c-index) des différentes politiques de sélection du modèle final (GBSG-2).

Aucune politique ne semble significativement meilleure que le choix aléatoire. Nous avons retenu le choix du logrank maximum. Dans une application concrète, avec l'avis d'un expert, il est possible de construire des modèles qui favorisent l'usage de certains attributs ou qui minimisent le nombre d'attributs différents présents dans le modèle.

3.4.6 Comparaison avec la littérature

Nous comparons maintenant la méthode des familles de motifs avec les méthodes de la littérature pour l'analyse de temps de survie. Les résultats sur les bases de données GBSG-2 et PBC sont présentés dans le Tableau 3.8. Les performances sont significativement meilleures que celles des arbres de décision. Les forêts de survie restent les plus performantes mais utilisent un modèle plus complexe composé de mille arbres. Enfin les résultats restent comparables avec l'ACDS. Comme indiqué dans le Tableau 3.6, les modèles des familles de motifs sont constitués d'en moyenne 11 motifs contre 15 pour la méthode classique. Dans [57] les modèles générés sont plus compacts (6 motifs en moyenne) mais avec des performances plus faibles (c-index de 0.6777 ± 0.023 sur la base d'entraînement). De plus nos motifs sont de degré inférieur à 4 alors que l'ACDS fournit des motifs plus complexes [57].

	GBSG-2	PBC
Arbre de survie	0.6253 ± 0.037	0.7497 ± 0.054
Forêt de survie	0.6909 ± 0.029	0.8369 ± 0.042
ACDS	0.6818 ± 0.029	0.8083 ± 0.044
Famille de motifs	0.6832 ± 0.029	0.8138 ± 0.029

Tableau 3.8 – Performances (c-index) face aux méthodes de la littérature sur la base de test.

3.5 Conclusion

Les problèmes d'analyse de temps de survie sont proches des problèmes de régression avec pour certaines observations une information partielle sur le temps de survie. La prise en compte de ces observations est le principal challenge de l'analyse de temps de survie. Nous proposons dans ce cadre une méthode reposant sur l'analyse combinatoire de données et sur la notion de famille de motifs. Cette méthode consiste en une génération exhaustive des motifs, puis en un regroupement des motifs en familles et une sélection d'un modèle décrivant les observations intéressantes.

La méthode proposée a des performances comparables à la littérature sur les deux bases considérées tout en fournissant un modèle simplement interprétable. Ses performances dépassent celles des modèles simples tels que les arbres de survie et s'approchent des modèles complexes des forêts de survie. D'autre part la présence de familles permet d'identifier des groupes d'observations ayant des propriétés de survie similaires. Ce regroupement permettrait dans le cas d'une étude pratique d'apporter plus de connaissances sur les observations. L'équivalence des motifs d'une même famille permet de construire simplement des modèles avec des objectifs plus complexes : minimiser le nombre d'attributs différents, minimiser le poids du modèle si on suppose une pondération sur les attributs etc.

Une piste d'étude à explorer concerne la génération des motifs. La méthode précédente [57] proposait une génération gloutonne et laissait peu de contrôle sur les motifs en sortie. Nous proposons une génération exhaustive qui permet de parcourir un espace plus grand de motifs tout en gardant un contrôle sur la couverture, le degré et la distance par rapport au comportement de survie moyen. En pratique le nombre de motifs générés est encore trop grand et une étape de filtrage est nécessaire pour pouvoir appliquer notre méthode en temps raisonnable.

La mise en situation sur des données originales de la méthode permettrait d'apporter quelques améliorations sur la construction du graphe en introduisant des contraintes métier, en particulier pour la mesure de similarité. Une similarité plus pertinente pour l'expert permettrait d'utiliser une version pondérée des algorithmes de détection de communautés.

Chapitre 4

Partition en graphes denses

4.1 Introduction

La détection de communautés[1] est une problématique fréquemment rencontrée lors de l'analyse de données. Ces données peuvent être de nature très variée : réseaux sociaux, images, maillages de structures, graphes, etc. Pour chaque paire de données, nous disposons d'une mesure de distance ou de similarité. Ces réseaux ont la particularité de présenter des structures de communautés, c'est-à-dire des ensembles E d'observations fortement similaires deux à deux mais différentes des observations hors de E.

L'exemple classique de détection de communautés est le réseau social du *karate club* étudié par Zachary [94]. Suite à un conflit, ce club de karaté a connu une séparation en deux nouveaux clubs, chaque membre allant dans le même club que ses amis. Les relations entre les membres du club sont représentées par le graphe Figure 4.1, la couleur des sommets donne la répartition des membres dans les deux nouveaux clubs. Ces deux ensembles présentent une structure de communautés avec des liens forts à l'intérieur et peu de liens vers l'extérieur. Pour d'autres exemples d'applications de la détection de communautés, le lecteur pourra se référer à l'article de Newman [68].

L'intérêt de la détection de communautés ne s'arrête pas à l'analyse de réseaux sociaux. En traitement d'image, un problème similaire consiste à isoler les différents objets composant l'image. Ces problèmes sont connus sous le nom de segmentation d'image. Shi et Malik [85] proposent de transformer les problèmes de segmentation d'images en partition de graphes complets pondérés. Chaque pixel de l'image correspond à un sommet du graphe et les arêtes ont pour poids une mesure de similarité entre les sommets. Les auteurs proposent différentes mesures de similarité ainsi qu'un critère d'évaluation de la partition appelé *normalized cut*. Dans ce cas l'objectif est de trouver des groupes de pixels avec une forte similarité mais étant distants des

1. Ce travail a été soumis au journal *Discrete Applied Mathematics*

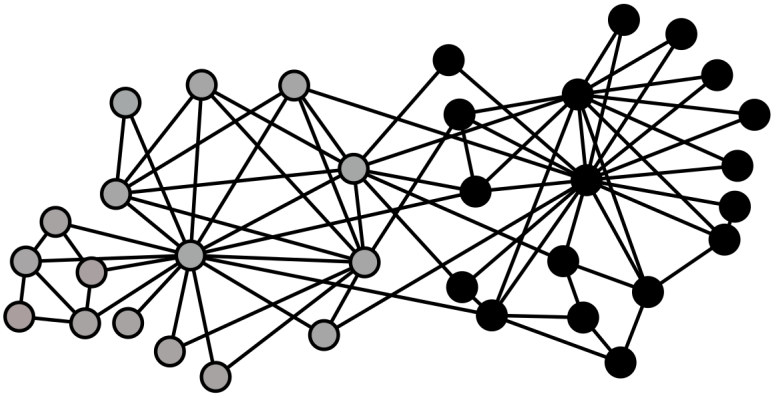

Figure 4.1 – Le graphe représentant le réseau social *karate club*. Les couleurs donnent la répartition des sommets dans les deux classes.

sommets à l'extérieur des groupes.

4.1.1 Fonctions objectif

Ces problèmes peuvent se modéliser comme des problèmes de partition des sommets d'un graphe maximisant un certain critère. Nous nous limiterons dans ce chapitre à des problèmes de partition ; dans la littérature les problèmes de couverture et les problèmes hiérarchiques ont aussi été étudiés [21].

Il existe dans la littérature de nombreuses fonctions objectif sans qu'aucune ne fasse vraiment l'unanimité. Ceci peut être expliqué par le théorème d'impossibilité de Kleinberg [52] qui stipule la chose suivante. Considérons un graphe G_d dont les arêtes sont pondérées par une pseudo-distance d et f une fonction qui à G_d associe une partition de ses sommets en communautés "proches" selon d. Considérons de plus les trois propriétés suivantes sur f :

- **Invariance** : si les poids sont multipliés par une constante positive a alors on obtient un graphe G_{ad} tel que $f(G_d) = f(G_{ad})$.
- **Complétude** : pour tout graphe G, et pour toute partition P des sommets de G, il existe une pseudo-distance d sur ses arêtes telle que $f(G_d) = P$.
- **Consistance** : si on a $f(G_d) = P$ et que d est légèrement modifiée de sorte à obtenir d' telle que deux sommets d'une même classe de P deviennent plus proches et que deux sommets de classes différentes deviennent plus éloignés alors $f(G_d) = f(G_{d'})$.

Ces propriétés semblent naturelles pour la détection de communautés, mais le théorème de Kleinberg dit qu'il est impossible de trouver une fonction f vérifiant les trois propriétés précédentes dès qu'on considère des graphes à au moins deux sommets.

Parmi les fonctions objectif de la littérature, on retiendra en particulier les critères *normalized cut*, *ratio cut*, *ratio association* proposés dans la communauté de la segmentation d'image [30, 85]. Ces fonctions sont en général difficiles (au sens de la complexité) à optimiser mais donnent de bons résultats en pratique [85, 91].

D'autres critères sont issus de l'optimisation combinatoire, comme le critère *min cut* [87], la force des graphes proposée par Cunningham [26] et d'autres fonctions issues de problèmes pratiques qui s'en rapprochent [76]. Pour ces problèmes des algorithmes polynomiaux existent, ils reposent sur des concepts théoriques forts tels que la minimisation de fonctions sous-modulaires et la partition de matroïdes. D'autres mesures bien connues en optimisation combinatoire sont NP-difficiles à optimiser, c'est le cas des *sparsest cut* et *max cut* [35]. Enfin des fonctions objectif basées sur différentes notions de densité ont été proposées mais restent NP-difficiles [86] à optimiser.

La théorie spectrale des graphes concerne l'étude des graphes par les valeurs propres et vecteurs propres des matrices d'adjacence ou du laplacien [20]. Les problèmes de partitionnement sont aussi étudiés du point de vue spectral, notamment avec l'utilisation du *Fiedler vector*.

La détection de communautés a proposé plusieurs mesures permettant d'évaluer la qualité d'une partition. La plus populaire est probablement la modularité proposée par Newman [71]. Cette mesure compare la distribution des arêtes à l'intérieur de chaque classe de la partition par rapport à la distribution attendue dans un graphe aléatoire ayant la même distribution de degrés. Du point de vue de la complexité, l'optimisation de ce critère est aussi un problème NP-difficile [16].

4.1.2 Techniques de résolution

Nous venons de voir qu'il existe de nombreuses fonctions objectif permettant d'évaluer une partition d'un graphe. Il est en général difficile de trouver la solution optimale à ces problèmes d'optimisation en temps polynomial. Nous allons voir quelles sont les principales heuristiques de résolution de ces problèmes difficiles. Pour une présentation plus détaillée, le lecteur pourra se référer à l'état de l'art de Schaeffer [83]. La plupart des mesures que nous avons présentées sont issues de problèmes pratiques pour lesquels des algorithmes sont proposés. Nous pouvons distinguer trois types d'approches classiques : les méthodes divisives qui vont récursivement séparer le graphe en plusieurs classes, les méthodes hiérarchiques qui vont construire petit à petit des groupes et les méthodes globales qui donnent directement une

partition en classes.

L'heuristique de Kernighan Lin [50] est une méthode divisive classique. Elle partitionne un graphe en effectuant une série d'échanges locaux. Les algorithmes classiques de flot maximum et coupe minimum [84, 87] sont aussi utilisés dans les méthodes divisives mais ont l'inconvénient de donner des partitions déséquilibrées (par exemple un sommet contre le reste du graphe). Les méthodes spectrales [20] permettent d'obtenir des bipartitions d'un graphe à partir du spectre de la matrice d'adjacence ou du laplacien du graphe. Elles ont été utilisées pour optimiser les critères *ratio cut, normalized cut, ratio association* [85, 91] et la modularité [70].

Les méthodes hiérarchiques ou accumulatives reposent sur l'idée qu'il y a différentes partitions possibles selon le niveau de granularité. Ces méthodes sont souvent gloutonnes, elles partent avec un sommet par classe et fusionnent les classes qui amènent la plus grosse augmentation de la fonction objectif. Newman propose un tel algorithme pour la modularité [69] qui s'avère efficace en pratique. Les différentes fusions effectuées par ces algorithmes sont représentées par un dendrogramme. La Figure 4.2 présente les différents regroupement effectués pour le graphe de la Figure 4.1 en utilisant l'algorithme de Newman [69].

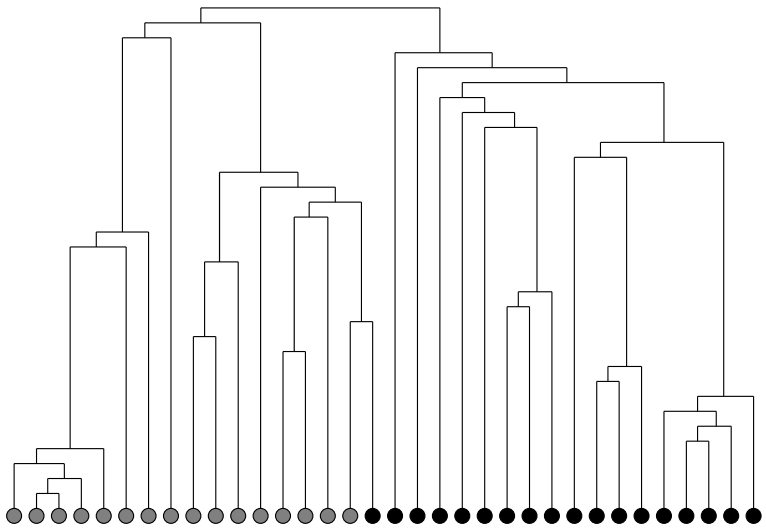

Figure 4.2 – Exemple de dendrogramme pour la partition du graphe *karate club* à l'aide de la modularité. Les deux classes sont représentées par les couleurs des cercles.

L'algorithme du *k-means* [45] est probablement l'algorithme de partition global le plus populaire. Il a été conçu à l'origine pour traiter des problèmes géométriques mais peut être facilement appliqué à des graphes pondérés. L'idée générale consiste à regrouper un ensemble de points dans un espace métrique en k classes telle que la somme des distances entre les

sommets d'une classe et son centre soit minimisée. Le problème général est NP-difficile [4, 28]. Une heuristique classique consiste, à partir d'un ensemble de k points (représentant les k classes), à alterner deux opérations : affecter les données à la classes la plus proche et calculer les k nouveaux centres à partir de l'affectation. Cet algorithme à l'origine géométrique a été adapté aux critères *ratio cut, normalized cut, ratio association* [30].

Les méthodes exactes de la recherche opérationnelle ont été utilisées pour optimiser la modularité. Brandes *et al.* [16] ont proposé un programme linéaire en nombres entiers pour trouver la partition maximisant la modularité d'un graphe. D'après nos expérimentations, cette formulation ne permet pas de traiter des graphes trop grands (plus de 20 sommets). Une formulation à base de génération de colonnes a été proposée par Aloise *et al.* [3], elle permet de traiter des graphes allant jusqu'à 500 sommets.

Dans ce chapitre, nous allons étudier le problème consistant à maximiser une fonction objectif que nous appelons densité. Ce critère est connu dans la littérature du traitement d'image sous le nom *ratio association* [30]. Trouver le sous-graphe le plus dense est un problème admettant un algorithme polynomial. Cette propriété nous semblait *a priori* intéressante pour construire des algorithmes d'approximation ou même exacts, la complexité de l'optimisation de la densité n'étant à notre connaissance pas établie dans la littérature. Nous montrons que ce problème est NP-difficile et nous proposons un algorithme polynomial pour les arbres. La Section 4.2 donne une définition formelle de la densité et quelques rappels de théorie des graphes. Dans la Section 4.3, nous montrons que le problème de partition associé à cet objectif est difficile. Enfin, un algorithme polynomial pour la classe des arbres est proposé dans la Section 4.4.

4.2 Définitions et premiers résultats

4.2.1 Résultats classiques de théorie des graphes

Dans cette section, nous rappelons quelques résultats classiques de la théorie des graphes nécessaires dans la suite. Les preuves ainsi que d'autres notions complémentaires peuvent être trouvées dans le livre de Diestel [31]. Nous travaillerons dans le cadre des graphes simples, non pondérés.

Un **couplage** est un sous-ensemble d'arêtes deux à deux disjointes. Un sommet est **couvert** par un couplage s'il est adjacent à une arête d'un couplage. Un couplage est dit **parfait** si tous les sommets du graphe sont couverts par le couplage. Un **transversal** de G est un sous-ensemble des sommets tel que chaque arête de G est adjacente à au moins un sommet du transversal. Dans un graphe G admettant un couplage M, une chaîne $P = (V, E)$ est dite **alternée** si $E \backslash M$ est un couplage. Une chaîne alternée est dite **augmentante** si ses extrémités ne sont pas couvertes par le couplage. Les

deux résultats suivants sont des résultats classiques en théorie des couplages (preuves omises) :

Théorème 4.1 (König [53])
Soient G un graphe biparti, M^* un couplage maximum et T^* un transversal minimum alors $|M^*| = |T^*|$.

Théorème 4.2 (Petersen [84])
Soit G un graphe, un couplage M est maximum si et seulement si il n'existe pas de chaîne augmentante.

4.2.2 Définitions et résultats sur la densité

Soit $G = (V, E)$ un graphe simple, connexe non orienté ; la densité de G est donnée par le rapport entre $|E|$ et $|V|$. Le terme densité est utilisé pour de nombreux ratios entre arêtes et sommets, en particulier une autre définition de la densité est proposée par Szemerédi [31].

Définition 4.1. La **densité** d'un graphe G est donnée par le rapport :

$$d(G) = \frac{|E|}{|V|}$$

Soit X un sous-ensemble de sommets de V, on note $E(X)$ l'ensemble des arêtes de E dont les deux extrémités sont dans X. Le sous-graphe induit par X est noté $G[X] = (X, E(X))$. Le problème d'optimisation classique lié à la densité consiste à trouver le sous-graphe le plus dense. Ce problème est polynomial et peut être résolu par des techniques de flots [36] ou par la programmation linéaire [18]. En ajoutant une contrainte sur le nombre de sommets du sous-graphe, le problème devient directement NP-difficile en généralisant le problème Max Clique.

Soit Π l'ensemble des partitions des sommets de G, la densité d'une partition est définie comme la somme des densités des graphes induits par chaque classe de la partition.

Définition 4.2. La **densité d'une partition** $P \in \Pi$ d'un graphe $G = (V, E)$ est donnée par :

$$d_G(P) = \sum_{X \in P} d(G[X]) = \sum_{X \in P} \frac{|E(X)|}{|X|}$$

Lorsqu'il n'y a pas de confusion sur le graphe à partitionner, nous utiliserons la notation $d(P)$ pour la densité de P. Afin d'alléger les notations, nous utiliserons $d(X)$ pour représenter la densité du graphe induit par X.

Nous noterons $\overline{G}$ le graphe complémentaire de $G = (V, E)$. La densité d'une partition P de G peut être reliée à la densité de la même partition dans $\overline{G}$.

Proposition 4.1
Soit P une partition de $G = (V, E)$ alors :

$$d_G(P) = \frac{|V|}{2} - \frac{|P|}{2} - d_{\overline{G}}(P)$$

Preuve. En utilisant la définition de la densité d'une partition et en notant $\overline{E}$ les arêtes de $\overline{G}$:

$$\begin{aligned} d_G(P) = &\sum_{X \in P} \frac{|E(X)|}{|X|} = \sum_{X \in P} \frac{\frac{|X|(|X|-1)}{2} - |\overline{E}(X)|}{|X|} = \sum_{X \in P} \frac{|X|-1}{2} - \frac{|\overline{E}(X)|}{|X|} \\ = &\frac{|V|}{2} - \frac{|P|}{2} - d_{\overline{G}}(P) \end{aligned}$$

□

Définition 4.3. L'**éparsité**[1] $F_G(P)$ d'une partition $P \in \Pi$ d'un graphe $G = (V, E)$ est définie par :

$$F_G(P) = \frac{|P|}{2} + d_G(P)$$

Dans la suite de ce chapitre nous nous intéresserons aux problèmes de décisions suivants :

Partition en graphes denses
Instance : Un graphe $G = (V, E)$ et un rationnel positif D
Question : Existe-t-il une partition $P \in \Pi$ telle que $d(P) \geq D$?

Partition en graphes épars
Instance : Un graphe $G = (V, E)$ et un rationnel positif D
Question : Existe-t-il une partition $P \in \Pi$ telle que $F(P) \leq D$?

k-Coloration
Instance : Un graphe $G = (V, E)$
Question : Existe-t-il une partition $P \in \Pi$ en k classes et telle que pour tout $X \in P$, $G[X]$ soit un ensemble stable ?

Les problèmes d'optimisation associés à ces problèmes de décisions seront précédés du terme Min ou Max. Par exemple :

Max Partition en graphes denses
Instance : Un graphe $G = (V, E)$
Solution : Une partition $P^* \in \Pi$ telle que $d(P^*) \geq d(P)$, pour tout $P \in \Pi$

1. Traduction libre de l'anglais *sparsity*

Proposition 4.2
Le problème d'optimisation MAX PARTITION EN GRAPHES DENSES sur un graphe G est équivalent au problème d'optimisation MIN PARTITION EN GRAPHES ÉPARS sur le graphe $\overline{G}$.

Preuve. Il s'agit d'une application directe de la Proposition 4.1. □

Proposition 4.3
Soit χ_G le nombre chromatique d'un graphe $G = (V, E)$ et P^* une partition de V minimisant F_G alors :

$$F_G(P^*) \leq \frac{\chi(G)}{2}$$

Preuve. Soit P la partition des sommets de G associée à une coloration en $\chi(G)$ couleurs où chaque classe de P correspond à une couleur de G. Les sommets de chaque classe de P forment un ensemble stable de densité nulle et donc $F(P) = \frac{\chi(G)}{2}$. Par définition $F(P^*) \leq F(P)$ et l'inégalité est valide. □

On remarquera que l'inégalité n'est pas toujours serrée. Par exemple pour un cycle à 5 sommets, la coloration optimale utilise 3 couleurs mais il existe une partition des sommets en deux classes avec $F(P) = 1 + \frac{1}{3} < \frac{3}{2}$ (voir Figure 4.3).

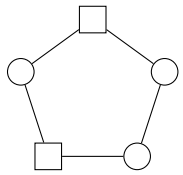

Figure 4.3 – Une partition du cycle à 5 sommets avec une éparsité inférieure à 3/2. Les classes des sommets sont représentées par des cercles et des carrés.

4.3 Complexité et approximabilité

Dans cette section nous allons montrer que le problème PARTITION EN GRAPHES DENSES est NP-complet. Dans un premier temps, nous étudierons le cas particulier où le nombre de classes de la partition est fixé, puis nous étudierons le cas général.

Lorsque le nombre de classes est spécifié, la Proposition 4.4 nous donne une réduction depuis le problème de k-COLORATION dans le graphe complémentaire.

Proposition 4.4
Dans un graphe $G = (V, E)$, les propositions suivantes sont équivalentes :

1. Il existe une k-coloration de $\overline{G}$,
2. Il existe une partition P des sommets en k classes telle que $F_{\overline{G}}(P) \leq \frac{k}{2}$.

Preuve. Montrons d'abord que $2 \Rightarrow 1$. Soit P une partition de V en k classes telle que $F_{\overline{G}}(P) \leq \frac{k}{2}$. Supposons par l'absurde qu'il existe une classe X de P telle que $\overline{G}[X]$ ne soit pas un stable. Le sous-graphe $\overline{G}[X]$ contient alors une arête et sa densité est strictement positive par définition. Pour toute classe Y de P différente de X, la densité de $\overline{G}[Y]$ est supérieure ou égale à 0. En utilisant la définition de l'éparsité d'une partition et les inégalités précédente on montre alors que $F_{\overline{G}}(P) > \frac{k}{2}$ ce qui est une contradiction.

La direction $1 \Rightarrow 2$ est donnée par la Proposition 4.3.

□

Dans le cas plus général où le nombre de classes est libre, la réduction depuis le problème de k-Coloration n'est pas immédiate. Nous allons d'abord montrer que le problème Partition en graphes épars est NP-complet en faisant une réduction depuis k-Coloration. Puisque les deux problèmes d'optimisation sont équivalents, le problème Partition en graphes denses sera lui aussi NP-complet.

L'idée générale pour montrer que Partition en graphes épars est NP-complet consiste à transformer une instance de k-Coloration en ajoutant des arêtes au graphe de façon à augmenter sa densité sans changer la qualité d'une solution correspondant à une k-coloration. De cette manière, les solutions du nouveau graphe vont induire une k-coloration du graphe initial.

On définit G^q le graphe construit depuis un graphe G où chaque sommet v est remplacé par un ensemble stable de taille q noté $\{v_1, ..., v_q\}$. Chaque arête (i, j) est remplacée par le graphe biparti complet $K_{q,q}$. Par exemple, le graphe C_5 présenté Figure 4.3 est transformé en le graphe C_5^2 de la Figure 4.4.

Une première observation nous permet d'affirmer que cette transformation ne change pas le nombre chromatique.

Proposition 4.5
Soient $\chi(G)$ et $\chi(G^q)$ les nombres chromatiques de G et G^q, alors $\chi(G) = \chi(G^q)$.

Preuve. L'inégalité $\chi(G^q) \leq \chi(G)$ est directe, puisque toute coloration de G reste une coloration de G^q. Supposons maintenant que $\chi(G^q) < \chi(G)$, alors en gardant un sommet de chaque ensemble $\{v_1...v_q\}$, nous obtenons une coloration de G en strictement moins de $\chi(G)$ couleurs ce qui est une contradiction. On a donc $\chi(G^q) \geq \chi(G)$ et l'égalité est montrée. □

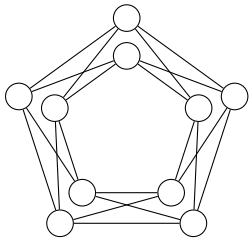

Figure 4.4 – Le graphe C_5^2.

Grâce à cette observation, nous pouvons prouver le résultat de complexité suivant.

Théorème 4.3
Le problème PARTITION EN GRAPHES ÉPARS est NP-complet.

Preuve. Le problème est clairement dans NP, puisque pour une partition P donnée, il est possible en temps polynomial de calculer $F(P)$.

Considérons une instance G du problème K-COLORATION. Nous pouvons supposer sans perte de généralité que G a plus de k sommets. Nous transformons G en une instance de PARTITION EN GRAPHES ÉPARS comme suit.

À partir d'un graphe G à n sommets, nous construisons le graphe G^q avec $q = n^4$. Nous allons montrer qu'il existe une k-coloration de G si et seulement si il existe une partition P de G^q telle que $F(P) \leq \frac{k}{2}$.

Si G admet une k-coloration, alors G^q admet aussi une k-coloration (Proposition 4.5). Par la Proposition 4.3, il existe une partition P des sommets de G^q telle que $F(P) \leq \frac{k}{2}$.

Supposons maintenant qu'il existe une partition P de G^q telle que $F(P)$ soit inférieure ou égale à $\frac{k}{2} \leq \frac{n}{2}$. Ceci implique que $|P| \leq k$. Considérons maintenant un sommet i de G quelconque et l'ensemble de sommets $I = \{i_1, ..., i_q\}$. Soit C_I la classe de P qui contient le plus grand nombre de sommets de I. On note S_I les sommets de I appartenant à C_I. Puisque $|P| \leq k$, on a $|S_I| \geq \frac{q}{k} \geq \frac{q}{n} \geq n^3$. Considérons maintenant une arête (i, j) de G, ainsi que les ensembles C_I et C_J.

Supposons un instant que $C_I = C_J$. Le nombre total de sommets dans C_I est inférieur ou égal au nombre de sommets dans G^q, à savoir $qn = n^5$. Le nombre d'arêtes induites par C_I est au moins le nombre d'arêtes entre S_I et S_J d'où $|E(C_I)| \geq n^3.n^3$. On a donc $F(P) \geq \frac{|E(C_I)|}{|C_I|} \geq n \geq k > \frac{k}{2}$ ce qui contredit notre hypothèse $F(P) \leq \frac{k}{2}$.

Ainsi, pour chaque arête (i, j) de G, les ensembles S_I et S_J appartiennent à des classes différentes de P. Il est donc possible de construire une k-coloration de G à partir de la partition P de G^q et des ensembles S_U pour tout sommet u. Chaque sommet i de G est colorié avec la couleur

correspondant à la classe C_I. □

En modifiant légèrement la preuve précédente, il est possible de montrer que le problème MIN PARTITION EN GRAPHES ÉPARS n'est pas approximable à un facteur constant $r > 1$.

Théorème 4.4
Il n'existe pas d'algorithme polynomial d'approximation avec un facteur r constant ($r > 1$) pour le problème MIN PARTITION EN GRAPHES ÉPARS, à moins que $P = NP$.

Preuve. Supposons par l'absurde qu'il existe un algorithme polynomial qui, pour un graphe G, retourne une partition P des sommets telle que $F(P) \leq rF(P^*)$, avec P^* une partition qui minimise F. En utilisant la même preuve que celle du Théorème 4.3 avec $q = rn^4$, nous pouvons obtenir une r-approximation du nombre chromatique en temps polynomial.

Considérons une arête (i, j) d'un graphe G et supposons que les classes C_I et C_J soient les mêmes dans une partition P de G^q. Alors $F(P) \geq d(C_I) \geq rn > r\frac{\chi(G^q)}{2} \geq rF(P^*)$ ce qui est une contradiction. On a donc S_I et S_J qui appartiennent à des classes différentes de P et il est possible de construire une coloration propre de G avec $|P|$ couleurs. Or $|P| \leq 2F(P) \leq 2rF(P^*) \leq r\chi(G^q) = r\chi(G)$. Puisque déterminer le nombre chromatique n'est pas approximable à un facteur constant [7], le problème MIN PARTITION EN GRAPHES ÉPARS n'est pas non plus approximable à un facteur constant. □

Ce dernier théorème ne permet pas de prouver qu'il n'existe pas d'algorithme d'approximation à facteur constant pour le problème MAX PARTITION EN GRAPHES DENSES. Pour cela il faudrait une réduction qui préserve l'approximabilité entre les problèmes de partition en graphes denses et de partition en graphes épars. D'après la Proposition 4.1, un terme proportionnel au nombre de sommets apparaît dans la relation qui lie la densité avec l'éparsité du graphe complémentaire et ne permet pas d'utiliser le passage au graphe complémentaire dans la réduction.

Ces résultats de complexité et d'inapproximabilité ne signifient pas que le problème ne peut être résolu en pratique. Une heuristique basée sur l'algorithme du *k-means* est présentée dans l'article [30] où la densité d'une partition est nommée *ratio association*.

4.4 Cas particulier des arbres

Nous allons maintenant regarder le problème de partition en graphes denses dans la classe particulière des arbres. D'un point de vue pratique, cette classe n'est pas très proche des instances que nous pouvons rencontrer

mais elle présente des intérêts théoriques en rapprochant le problème de partition des problèmes bien connus de couplages et de transversaux. Notre approche consiste à donner une caractérisation des solutions optimales du problème MAX PARTITION EN GRAPHES DENSES puis nous proposons un algorithme polynomial trouvant la solution optimale. Cet algorithme utilise un paradigme de programmation dynamique sur un arbre enraciné.

4.4.1 Bornes sur la densité

Proposition 4.6
Soit $T = (V, E)$ un arbre avec au moins un sommet, sa densité d est donnée par :

$$d(T) = \frac{m}{n} = \frac{n-1}{n} = 1 - \frac{1}{n}$$

À partir de cette définition nous pouvons en déduire une borne inférieure de $\frac{1}{2}$ (pour les arbres avec au moins deux sommets) et une borne supérieure de 1 pour la densité d'un arbre. La borne inférieure est atteinte lorsque T est une arête, alors que la borne supérieure n'est jamais atteinte.

Plaçons nous maintenant dans le cadre plus général des graphes bipartis pour en déduire des bornes sur la densité d'une partition.

Proposition 4.7
Soit $G = (V_1 \cup V_2, E)$ un graphe biparti, alors $d(G) \leq \frac{n}{4}$.

Preuve. En utilisant la définition de la densité :

$$d(G) = \frac{m}{n_1 + n_2} \leq \frac{n_1 n_2}{n_1 + n_2} \leq \frac{n_1 + n_2}{4}$$

La première inégalité vient du fait que G ne peut pas contenir plus d'arêtes que le graphe biparti complet alors que la seconde vient de l'inégalité :

$$(n_1 + n_2)^2 - 4n_1 n_2 = (n_1 - n_2)^2 \geq 0$$

□

Puisque chaque sous-graphe d'un graphe biparti est lui même biparti, nous pouvons en déduire une nouvelle borne supérieure sur la densité d'une partition optimale.

Proposition 4.8
Soient $G = (V_1 \cup V_2, E)$ un graphe biparti et P^* une partition optimale. Alors $d(P^*) \leq \frac{n}{4}$.

Preuve. En appliquant la définition de la densité d'une partition :

$$d(P^*) = \sum_{X \in P^*} d(G[X]) \leq \sum_{X \in P^*} \frac{|X|}{4} = \frac{n}{4}$$

□

On remarquera que cette borne est serrée si G admet un couplage parfait. Cette borne permet de montrer un premier lien entre partition optimale et couplage dans les graphes bipartis.

4.4.2 Structure d'une partition optimale

Avant de revenir au cas des arbres, nous donnons un résultat de connexité dans les classes d'une partition optimale d'un graphe quelconque.

Proposition 4.9
Soient $G = (V, E)$ un graphe connexe et P^* une partition optimale en graphes denses. Alors pour toute classe $X \in P^*$, $G[X]$ est connexe.

Preuve. Par l'absurde, si $G[X]$ n'est pas connexe dans une partition P, nous allons montrer qu'il est possible de construire une nouvelle partition P' en remplaçant la classe X par de nouvelles classes correspondant à chacune des k composantes connexes de $G[X]$. Cette nouvelle partition est strictement meilleure que P :

$$d(P') - d(P) = \left(\sum_{i=1}^{k} \frac{|E(X_i)|}{|X_i|} \right) - \left(\frac{\sum_{i=1}^{k} |E(X_i)|}{\sum_{i=1}^{k} |X_i|} \right) \geq 0$$

puisque pour tout $a_i \geq 0$, $b_i > 0$, nous avons $\sum_i \left(\frac{a_i}{b_i} \right) \geq \frac{\sum_i a_i}{\sum_i b_i}$. □

En plus de ce résultat général sur les graphes, nous pouvons montrer que dans le cas des arbres, le sous-graphe induit par chaque classe d'une partition optimale contient au moins une arête. Nous allons ensuite montrer que ces classes sont des étoiles.

Proposition 4.10
Soient $T = (V, E)$ un arbre et P^* une partition optimale de T, alors toute classe de P^* est constituée d'au moins deux sommets.

Preuve. Supposons par l'absurde qu'il existe une classe X de P^* telle que $X = \{u\}$ avec $u \in V$. Puisque T est connexe alors il existe $v \in V$ telle que (u, v) soit une arête. On notera C la classe de v dans P^*. Construisons maintenant l'ensemble de sommets $C' = C \cup X$. Puisque C est connexe, C' est aussi connexe. Comparons maintenant les densités des partitions P^* et P la partition issue de P^* où les classes C et X sont fusionnées :

$$d(P) - d(P^*) = \frac{|C'| - 1}{|C'|} - \frac{|C| - 1}{|C|} = \frac{|C|}{|C| + 1} - \frac{|C| - 1}{|C|} > 0$$

La partition P est donc strictement plus dense que P^* ce qui est une contradiction. □

Proposition 4.11
Soient T un arbre et P^* une partition optimale de T, alors pour toute classe X de P^*, le graphe $G[X]$ n'admet pas comme sous-graphe induit une chaîne à 4 sommets.

Preuve. Supposons par l'absurde qu'il existe une telle classe X et soient $\{v, w, x, y\}$ les sommets de la chaîne. En enlevant l'arête (w, x) de $G[X]$, nous créons deux composantes connexes X' et X''. Puisque $G[X]$, $G[X']$ et $G[X'']$ sont des arbres, nous avons $d(G[X]) < 1$ et $d(G[X']) + d(G[X'']) \geq 1$ et la partition P^* n'est pas optimale. □

Cette condition nous permet de déduire le corollaire suivant sur la structure d'une partition optimale :

Corollaire 4.1
Soient T un arbre et P^* une partition optimale en graphes denses, alors pour chaque chaque classe X de P^*, le graphe $G[X]$ est une étoile.

La Proposition 4.8 indique un lien entre les couplages parfaits et les partitions optimales d'un graphe biparti. Le Corollaire 4.1 semble indiquer que dans les arbres ce lien est encore plus fort comme nous le montrons dans le résultat suivant.

Proposition 4.12
Soient T un arbre, M^* un couplage maximum et P^* une partition optimale en graphes denses. Alors $|P^*| = |M^*|$.

Preuve. L'inégalité $|M^*| \geq |P^*|$ vient du Corollaire 4.1. Il est possible de construire un couplage M en choisissant une arête dans chaque classe de P^*. Montrons par l'absurde que M est maximum. Si M n'est pas maximum alors par le Théorème 4.2, il existe une chaîne augmentante. Soit C un chaîne augmentante minimum (en nombre de sommets) et soient u, v ses extrémités.

Montrons que u et v ne peuvent appartenir à la même classe de P^*. Si la chaîne C n'est pas réduite à l'arête (u, v), alors elle doit contenir au moins 4 sommets. Grâce à la Proposition 4.11, nous pouvons affirmer que la chaîne n'est pas contenue dans une seule classe et donc que u et v appartiennent à des classes différentes. Si C est réduite à l'arête (u, v), puisque u et v ne sont pas couverts par le couplage ils doivent appartenir à des classes différentes autrement cela contredit le Corollaire 4.1. Les extrémités de la chaîne appartiennent donc à des classes différentes de P^*.

La classe contenant u et la classe contenant v possèdent chacune au moins 3 sommets, sinon u et v seraient couverts par le couplage. Notons X_u

(resp. X_v) la classe de u (resp. v) et soit $x \in X_u$ (resp. $y \in X_v$) le sommet le plus proche de v (resp. u) dans C. Il est possible que $x = u$ ou $y = v$ mais puisque u et v appartiennent à des classes différentes, $x \neq y$. Notons a et b les voisins respectifs de u et v dans C. La Figure 4.5 propose un exemple de partition induisant un couplage qui n'est pas maximum (graphe du haut).

Puisque C est minimum, chaque sommet de $C \setminus \{a, b\}$ est de degré au plus 2 dans le sous-graphe induit par sa classe. En effet puisque u et v ne sont pas saturés par le couplage et que X_u, X_v induisent des étoiles, u et v sont des sommets de degré 1 dans le graphe induit par leur classe. Si parmi les sommets restants il en existe un de degré strictement plus grand que 2 alors il existe une chaîne augmentante plus petite que C et qui contredit sa minimalité.

Soit M' le couplage obtenu à partir de M en échangeant ses arêtes sur la chaîne augmentante. Nous créons maintenant une nouvelle partition P' à partir de P^* en retirant x de X_u, y de X_v. Chaque arête de C qui n'était pas dans M forme une nouvelle classe de P'. Les éventuels voisins de a (resp. b) qui appartenaient à X_a (resp. X_b) mais pas à C appartiennent maintenant à la nouvelle classe de a (resp. b). À titre d'exemple, la partition en 3 classes Figure 4.5 est transformée en une meilleure partition en 4 classes.

Les arêtes (u, a) et (v, b) sont dans le nouveau couplage M' et par conséquent la classe X'_a de a dans P' est différente de X'_b la classe de b dans P'. Les classes de la nouvelle partition P' induisent bien des étoiles.

Durant cette opération, les classes X_u et X_v ont chacune perdu un sommet et une nouvelle classe a été créée. La différence de densité entre la nouvelle partition P' et P^* est donnée par :

$$\Delta = d(P') - d(P^*) \geq \frac{1}{2} - \frac{1}{|X_u|(|X_u| - 1)} - \frac{1}{|X_v|(|X_v| - 1)}$$

Puisque $|X_u| \geq 3$ et $|X_v| \geq 3$, nous avons $\Delta > 0$ et P^* n'était pas optimal contrairement à l'hypothèse initiale. □

En utilisant le Théorème 4.1 qui donne un lien entre transversal minimum et couplage maximum dans les graphes bipartis, nous obtenons le corollaire suivant.

Corollaire 4.2
Soient T^* un transversal minimum et P^* une partition optimale en graphes denses, alors chaque classe de la partition contient exactement un sommet de T^*.

Preuve. Le Corollaire 4.1 nous permet d'affirmer que chaque classe de P^* est une étoile non triviale. Chaque classe de P^* contient donc au moins un sommet de T^*. En utilisant la Proposition 4.12 et le Théorème 4.1 nous avons :

$$|P^*| = |M^*| = |T^*|$$

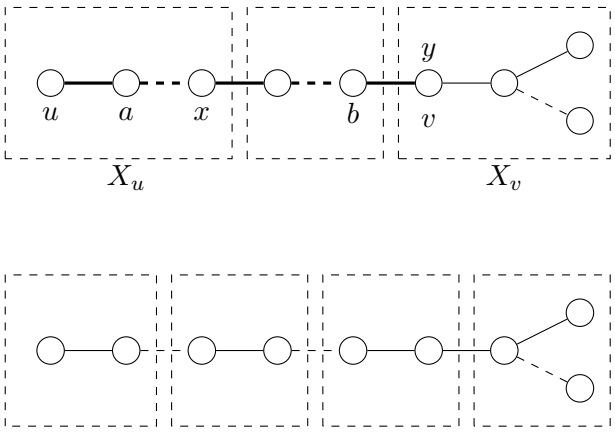

Figure 4.5 – Exemple de partition avec une chaîne augmentante (en gras sur le graphe du haut), les classes sont représentées par des rectangles en pointillés. Les arêtes du couplage sont représentées par des traits pointillés. Une meilleure partition (graphe du bas) est construite à partir de la chaîne augmentante.

Ces égalités nous permettent d'affirmer que chaque classe contient exactement un sommet de T^*. □

Les résultats obtenus dans cette section nous permettent de caractériser les partitions optimales dans un arbre en s'appuyant sur un transversal minimum. De plus trouver un transversal minimum dans un graphe biparti est un problème polynomial. Nous allons maintenant voir comment construire une partition optimale dans un arbre.

4.4.3 Algorithme de partition

Dans cette section, nous présentons un algorithme dynamique qui construit une partition optimale au problème MAX PARTITION EN GRAPHES DENSES pour les arbres. L'algorithme proposé est basé sur les algorithmes dynamiques classiques permettant de trouver le transversal minimum ou le stable maximum dans un arbre. Soit $T = (V, E, u)$ un arbre enraciné en u un sommet quelconque de V. Nommons T_i le sous-arbre enraciné constitué de i un fils de u et de ses descendants. Nous appelons F'_i la forêt induite par les sommets de $T_i \setminus \{i\}$. La Figure 4.6 présente une telle configuration.

L'idée principale consiste à construire la partition optimale de T en utilisant la partition optimale des arbres T_i et des forêts F'_i. Nous utiliserons aussi les propriétés structurelles précédentes, en particulier le lien entre un transversal minimum et partition optimale.

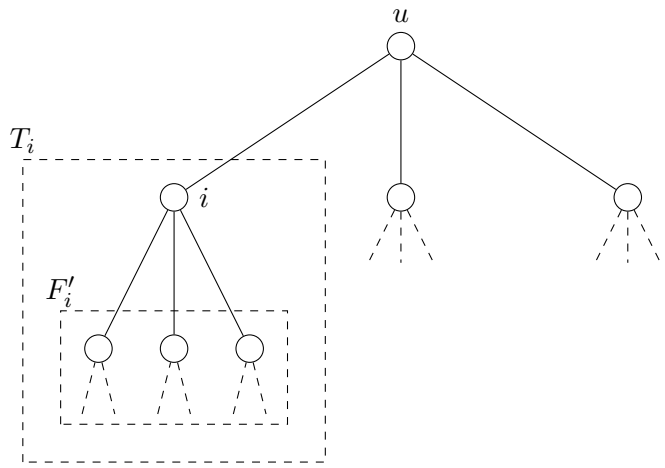

Figure 4.6 – Un arbre enraciné en u avec un fils i, un sous-arbre T_i et une forêt F'_i.

Théorème 4.5
Le problème MAX PARTITION EN GRAPHES DENSES admet un algorithme polynomial sur les arbres.

Preuve. Il est clair que si T est un arbre de hauteur un (une étoile), trouver la partition optimale de T est trivial. La forêt F' est alors constituée de sommets disjoints et trouver sa partition optimale est trivial.

Supposons que T soit de hauteur 3 ou plus et calculons T^* un transversal minimum. Enracinons T en un sommet u quelconque. Deux cas de figures se présentent : soit u est dans le transversal T^* soit il est en dehors. Dans la suite nous noterons W les fils de u.

Premier cas $u \in T^*$, alors u a deux types de fils : ceux qui appartiennent à T^* et les autres que nous noterons $W' = W \backslash T^*$. En utilisant le Corollaire 4.2, nous pouvons affirmer que u et les fils de u dans T^* ne peuvent être dans la même classe. Pour les fils i de u qui sont dans T^*, nous utilisons la partition optimale des arbres T_i correspondant.

Soit X_u la classe de u, il faut décider quels sont les sommets de W' qui seront dans X_u. Le cas des sommets isolés se règle rapidement, puisque d'après la Proposition 4.10 ils doivent être attachés à une classe et dans ce cas ils ne peuvent être rattachés qu'à la classe X_u.

Pour chaque fils de u restant i, soit Δ_i la différence entre la densité des partitions optimales de T_i et de F'_i. Nous trions les sommets restants par Δ_i croissants. Il est clair que dans une partition optimale P^* si un sommet j appartient à X_u alors tous sommets i tels que $\Delta_i < \Delta_j$ appartiennent

aussi à X_u, sinon en échangeant i et j nous créons une partition P telle que $d(P) - d(P^*) = \Delta_j - \Delta_i > 0$.

Ajoutons les sommets dans l'ordre des Δ_i croissants à X_u jusqu'au sommet j tel que $d(X_u \cup \{j\}) - d(X_u) < \Delta_j$. Pour ce sommet j et tous les suivants, les ajouter à X_u entraîne une diminution de la densité totale.

La partition optimale P^* est constituée de X_u, des partitions optimales des sous-arbres T_i pour $i \in W \cap T^*$, des partitions optimales des forêts F'_j pour $j \in X_u$ et des partitions optimales des arbres T_k pour $k \in W' \setminus X_u$.

Second cas $u \notin T^*$, alors chaque fils de u est dans T^*. Par la Proposition 4.10, u doit appartenir à la classe d'un de ses fils. Puisque u possède un nombre polynomial de fils, il est possible en temps polynomial de tester toutes les possibilités d'affectation de u et de garder la meilleure pour la partition de T.

Enfin pour obtenir la meilleure partition de la forêt $F' = T \setminus \{u\}$, il suffit de considérer la partition optimale de chaque arbre T_i pour i un fils de u. □

L'algorithme évoqué dans la preuve du Théorème 4.5 est décrit formellement dans l'Algorithme 1 sous le nom PGD. Il fait appel à deux fonctions correspondant aux deux cas de la preuve du Théorème 4.5. Ces deux fonctions, Affecter et AjouterFils sont décrites respectivement dans l'Algorithme 2 et dans l'Algorithme 3.

La complexité de l'algorithme PGD est en $O(n^2 \log n)$. En effet la fonction PGD est appelée sur chaque sommet de l'arbre. À chaque étape un transversal minimum est calculé sur un arbre avec une complexité $O(n)$ et une des deux procédures AjouterFils et Affecter est appelée. La procédure Affecter s'effectue en temps linéaire alors que la procédure AjouterFils s'effectue en temps $O(n \log n)$ puisqu'elle nécessite un tri des fils.

Le calcul d'un transversal minimum peut être effecuté par un algorithme dynamique dans les arbres. Ainsi en enrichissant l'Algorithme 1, l'étape de calcul du transversal peut s'effectuer en temps constant. D'autre part chaque sous arbre T_i n'est trié qu'une seule fois dans la procédure AjouterFils. La complexité de cet algorithme enrichi est alors $O(n \log n)$.

4.5 Conclusions

Dans ce chapitre nous avons présenté un problème d'analyse de données non supervisé : la détection de communautés. Nous avons décrit le concept de densité d'une partition. Nous utilisons cette densité comme fonction objectif d'un problème de partitionnement de graphes. D'un point de vue théorique le problème est NP-difficile et les résultats du Théorème 4.4 semblent indiquer qu'il sera difficile de trouver un algorithme d'approximation avec une

Algorithme 1 PGD(u)

si u est une feuille **alors**
  retourner $(\{\{u\}\}, \{\emptyset)\}$
finsi
$P \leftarrow \emptyset$, $P' \leftarrow \emptyset$
pour tout i un fils de u **faire**
  $(P_i, P_i') \leftarrow$ PGD(i)
  $P' \leftarrow P' \cup P_i$
fin pour
$T^* \leftarrow$ MinTransversal(T_u)
si $u \in T^*$ **alors**
  $X_u \leftarrow \{u\}$
  AjouterFils(X_u)
sinon
  Affecter(u)
finsi
retourner (P, P')

Algorithme 2 Affecter(u)

pour tout i un fils de u **faire**
  $P = P \cup P_i$
fin pour
$D \leftarrow d(P)$, $k \leftarrow \emptyset$
pour tout i un fils de u **faire**
  $X_i = X_i \cup \{u\}$
  si $d(P) > D$ **alors**
    $k \leftarrow i$, $D \leftarrow d(P)$
  finsi
  $X_i = X_i \setminus \{u\}$
fin pour
$X_k = X_k \cup \{u\}$

Algorithme 3 AjouterFils(X_u)

$P \leftarrow X_u$
pour tout i un fils de u **faire**
  $P = P \cup P_i$
fin pour
Trier les fils i de u selon Δ_i croissant
$i \leftarrow$ premier fils de u dans la liste ordonnée
tantque $d(X_u \cup \{i\}) - d(X_u) > \Delta_i$ **faire**
  $P = P - P_i + P'_i$
  $X_u = X_u \cup \{i\}$
  $i \leftarrow$ Successeur(i)
fin tantque

garantie de performance. Nous proposons un algorithme exact dans le cas où le graphe est un arbre. Cet algorithme repose sur un lien fort entre la structure d'une solution optimale et les couplages maximaux.

Une perspective immédiate consiste à étendre l'algorithme exact aux graphes bipartis puisque quelques résultats structurels restent vrais pour cette classe. D'un point de vue pratique, un algorithme exact ou d'approximation prenant en compte le poids des arêtes sur des graphes de type grille aurait des applications directes en traitement d'images. Les pixels d'une image sont représentés par les sommets d'une grille et les arêtes sont pondérées par la similarité de ses extrémités.

Une seconde perspective, un peu plus générale consiste à faire une étude comparative des différentes fonctions objectif proposées. Nous avons vu en introduction plusieurs de ces fonctions provenant de diverses communautés. Il serait bon d'un point de vue empirique d'avoir une étude comparative des différents objectifs afin de savoir quel objectif est plus adapté à l'application. Une telle comparaison demande d'implémenter des méthodes de résolutions exactes et efficaces afin d'évaluer la pertinence de chaque objectif sur des graphes classiques de la littérature. Des travaux dans ce sens ont été proposés par Aloise *et al.* [3] pour la modularité.

Le domaine de la détection de communautés propose des problèmes d'optimisation de grande taille avec des objectifs souvent difficiles au sens de la complexité. Les techniques classiques de recherche opérationnelle (approchées ou exactes) peuvent apporter de bonnes solutions pratiques à ces problèmes d'analyse de données.

Chapitre 5

Problèmes d'identification et sélection d'attributs

Le problème de sélection d'attributs [1] a déjà été mentionné tout au long de cette thèse. Il consiste à sélectionner dans une base de données Ω constituées de n observations, un sous-ensemble d'attributs pertinents parmi les m attributs présents en base. Une autre définition issue de [23] consiste à dire que la sélection d'attributs vise à diminuer le nombre d'attributs afin de faciliter l'apprentissage sans nuire à la qualité du modèle. On peut modéliser le problème de sélection d'attributs comme un problème d'optimisation qui cherche le sous-ensemble d'attributs maximisant une certaine fonction objectif.

Ce problème se pose en pratique lorsque la base de données contient un grand nombre d'attributs. Toutes ces informations ne sont pas nécessairement pertinentes et certaines peuvent être exclues de l'étude. Une première motivation est purement algorithmique : si l'instance est plus petite, le temps d'exécution de l'algorithme d'apprentissage diminue. Un tel gain n'est pas négligeable lorsque le nombre d'observations est grand et qu'il faut répéter l'apprentissage un grand nombre de fois, dans le cas d'une validation croisée par exemple. Une autre motivation vient du fait qu'un grand nombre d'attributs peut entraîner du sur-apprentissage en introduisant dans le modèle final des attributs dont le lien avec l'objectif à prédire n'est qu'une coïncidence. Enfin, la base de données peut contenir des attributs synonymes, décrivant le même phénomène. Conserver l'ensemble des synonymes favorise ce phénomène par rapport aux autres. On peut donc être amené à ne garder qu'un seul attribut parmi les attributs synonymes.

La sélection d'attributs se modélise comme un problème d'optimisation consistant à trouver le sous-ensemble d'attributs permettant de maximiser une certaine fonction objectif. Les techniques utilisées en pratique s'inspirent

1. Ce travail a été initié avec Sylvain Gravier, chercheur à l'Institut Fourier.

d'heuristiques classiques en recherche opérationnelle : algorithmes gloutons, algorithmes génétiques, *branch & bound*, recuit simulé, etc. [40]. On distingue deux grands types d'approches selon la nature de la fonction objectif qui va évaluer le sous-ensemble d'attributs.

Les *wrappers* embarquent un algorithme d'apprentissage et un critère d'évaluation pour cet algorithme. La fonction objectif utilisée pour la sélection d'attributs est alors la performance de l'algorithme sur le sous-ensemble d'attributs considéré. L'algorithme d'apprentissage utilisé dans le *wrapper* peut être différent de celui qui sera utilisé lors de la phase d'apprentissage. Des algorithmes moins performants mais plus rapides peuvent être utilisés dans la sélection d'attributs puisqu'il va falloir générer et évaluer de nombreux modèles.

Les filtres (*filters*) utilisent pour fonction objectif un critère d'évaluation indépendant de la méthode d'apprentissage. On peut citer, à titre d'exemple, des indicateurs statistiques tels que le coefficient de corrélation de Pearson ou le test du χ^2 entre l'objectif à prédire et l'attribut considéré. Les critères utilisés en pratique seront détaillés dans la Section 5.1.1.

Dans ce chapitre, nous nous intéressons aux approches de type filtre dans le cas des problèmes de classification. Ces approches ont l'avantage d'être indépendantes d'un algorithme d'apprentissage. Elles utilisent des fonctions objectifs plus simples qui permettent éventuellement une étude structurelle des problèmes. Plus particulièrement nous allons nous concentrer sur la classification binaire lorsque les attributs sont eux aussi binaires. Ce contexte fait apparaître une dimension combinatoire au problème de sélection d'attributs qui n'existe pas forcément dans le cas général. Nous présentons dans ce chapitre une vision combinatoire et algorithmique de la sélection d'attributs.

La Section 5.1 présente les méthodes classiques de la littérature et les méthodes combinatoires pour la sélection d'attributs. Dans la Section 5.2, nous proposons un problème combinatoire modélisant la sélection d'attributs binaires. Il consiste à identifier les couleurs des arêtes d'un hypergraphe muni d'une bicoloration de ses arêtes. L'identification se fait à partir de l'intersection des arêtes avec un sous-ensemble de sommets. La Section 5.3 présente une variante du problème de la Section 5.2 où l'identification se fait en comparant la cardinalité de l'intersection avec un seuil.

5.1 Filtres de la littérature

Les méthodes de la littérature pour la sélection d'attributs peuvent se diviser en deux grandes familles. D'un côté, les méthodes numériques consistent à trouver le sous-ensemble d'attributs maximisant un objectif donné. Elles sont assez générales pour s'appliquer à différents types de données et en général ne donnent pas de garantie d'optimalité. Les méthodes combinatoires s'appliquent pour les problèmes de classification avec des

données entières. Elles s'approchent de problèmes combinatoires bien connus comme la couverture par des tests [54]. Nous allons voir dans cette section les méthodes de la littérature pour ces deux grandes familles.

5.1.1 Méthodes numériques

Les méthodes de classement sont des algorithmes de sélection d'attributs qui traitent indépendamment chaque attribut en évaluant sa dépendance avec la classe à prédire. Cette évaluation peut se faire avec des outils statistiques tels que le coefficient de corrélation de Pearson, le test du χ^2, etc. [88]. D'autres critères sont issus de la théorie de l'information [40]. Les attributs sont alors classés selon le critère de dépendance et seuls les k premiers sont conservés. Cette méthode présente un défaut évident : elle ne tient pas compte des liens entre les attributs mais uniquement du lien avec la classe. Ainsi les k attributs conservés peuvent être tous identiques pour peu qu'ils soient très liés à la classe de l'observation. À l'opposé, une variable qui semble peu utile seule peut devenir très pertinente lorsqu'elle est combinée à d'autres [40].

L'Analyse en Composantes Principales (ACP) [92] est une méthode de sélection d'attributs non supervisée. Contrairement à la méthode de classement précédente, l'ACP prend uniquement en compte le lien entre les attributs pour sélectionner un sous-ensemble. Elle aura donc tendance à supprimer des redondances. Cette méthode a le défaut de ne pas tenir compte de la classe des observations en sélectionnant ses composantes principales.

D'autres méthodes tentent de prendre en compte le lien avec la classe et les liens entre les attributs. Le critère nommé CFS (*Correlation based Feature subset Selection* [41]) permet de trouver des sous-ensembles d'attributs fortement corrélés avec la classe mais faiblement corrélés entre eux.

Un autre critère, la mesure d'inconsistance proposée dans [29] permet d'évaluer la qualité d'un sous-ensemble d'attributs. Cette mesure compte le nombre d'observations avec des classes différentes qui sont inséparables pour le sous-ensemble d'attributs considéré. Les auteurs comparent différentes heuristiques (recherche exhaustive, algorithme de type Las Vegas, etc.) pour trouver un sous-ensemble minimal d'attributs sans inconsistance.

Des algorithmes de sélection d'attributs reposent sur la solution de programmes mathématiques [15]. Dans le cadre de la classification binaire, un programme linéaire est proposé pour séparer les observations de chaque classe par un hyperplan dans l'espace des attributs. Les attributs présents dans l'équation de l'hyperplan sont conservés comme étant les plus pertinents puisqu'ils permettent de séparer les observations des deux classes. Ces méthodes ont l'avantage de fournir une solution optimale en un temps raisonnable en utilisant les techniques de résolution classiques de programmes linéaires.

5.1.2 Méthodes combinatoires

Les méthodes combinatoires, à la différence des méthodes numériques précédemment décrites, ne permettent de traiter que des données avec des attributs nominaux. Dans le cas d'attributs numériques, il est alors nécessaire de binariser les données. Les méthodes qui sont détaillées dans cette section cherchent des sous-ensembles d'attributs minimaux qui permettent de séparer deux observations de classes différentes. Elles ne tiennent pas compte d'indicateurs statistiques externes mais uniquement des observations de la base de données. Elles sont par conséquent plus sensibles aux bruits (mauvais diagnostic, erreur de saisie, etc.).

Dans un premier temps, nous allons voir les solutions proposées à la sélection de support dans le cadre de l'analyse combinatoire de données. Dans un deuxième temps, nous nous intéressons à un problème combinatoire proche : la couverture par les tests.

Sélection de support Crama *et al.* [24] proposent une méthode de sélection d'attributs dans le cadre de l'analyse combinatoire de données. Un **support** est un sous-ensemble d'attributs tel qu'il n'existe pas d'observation de la classe 1 et de la classe 0 qui aient les mêmes valeurs d'attributs sur ce sous-ensemble. Un support minimal permet donc d'expliquer le concept (la classe à prédire), en enlevant les attributs redondants. Ce problème est présenté dans le Chapitre 2.

Couverture par des tests Le problème de couverture par des tests peut se définir de la manière suivante [54]. Étant donnée une matrice M dont toutes les lignes sont différentes, trouver un sous-ensemble de colonnes telles que la sous-matrice résultante ait ses lignes deux à deux distinctes. Il s'agit d'un problème de sélection de support pour un problème multiclasse où chaque classe contient exactement une observation.

Ce problème a des applications directes dans des domaines variés tels que le diagnostic, le test, la détection de pannes etc. [54, 74]. À titre d'exemple, considérons la matrice M suivante où chaque ligne représente une pathologie et chaque colonne un test.

$$M = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

L'élément M_{ij} vaut 1 si le test j est positif pour la pathologie i, 0 sinon. Puisque les lignes de la matrice sont toutes deux à deux distinctes, il est possible d'identifier chaque pathologie (a, b, c, d, e) à l'aide des six tests. La

pathologie a est, par exemple, la seule qui est positive pour l'ensemble des six tests. Remarquons que ces six tests ne sont pas tous nécessaires. En effet, la restriction M' de la matrice M aux colonnes 1, 2 et 4 permet d'obtenir une sous-matrice où les lignes sont toutes deux à deux distinctes :

$$M' = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 4 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array}$$

La pathologie b peut, par exemple, être identifiée comme la pathologie pour laquelle les tests 1 et 2 sont négatifs et le test 4 positif.

En attribuant un coût à chaque colonne, le problème d'optimisation qui se pose consiste à trouver un sous-ensemble de colonnes de coût minimal permettant d'identifier chaque ligne. Une solution d'un tel problème permet un diagnostic à moindre coût des diverses pathologies. Dans le cadre d'une application médicale, plusieurs pondérations peuvent être intéressantes : le coût financier, la pénibilité du test, etc. Ce problème est un problème NP-difficile, il est connu dans la littérature sous le nom de *minimum test set* [35]. Un cas particulier a été bien étudié, il s'agit du cas où M est la matrice d'adjacence d'un graphe. Dans ce cas on parle de problèmes de code identifiant [49].

Le problème que nous souhaitons traiter est plus général. En effet, nous n'avons pas besoin d'identifier chaque ligne mais seulement des groupes de lignes. Dans les applications qui nous intéressent, les lignes correspondent à des observations (typiquement des patients) et non à des pathologies. Il serait trop contraignant de demander à pouvoir identifier chaque observation à partir de ses attributs mais il reste envisageable de pouvoir discriminer des groupes d'observations (typiquement : les patients malades et les patients sains). Cette approche est équivalente au problème de sélection de support proposé par Crama *et al.* pour l'analyse combinatoire de données [24].

Nous venons de voir que les problèmes de sélection d'attributs généralisent les problèmes de couverture par des tests. Ces deux problèmes étant NP-difficiles, nous allons adopter une approche structurelle. Nous cherchons dans ces structures des bornes supérieures et inférieures sur la taille d'un sous-ensemble de colonnes permettant d'identifier les groupes de lignes. Nous utilisons comme support une structure classique en combinatoire : les hypergraphes.

5.2 Sélection d'attributs binaires

5.2.1 Modélisation

La base de données contenant les observations, leurs attributs et leur classe peut être représentée comme un hypergraphe muni d'une coloration des arêtes. Les sommets correspondent aux attributs et chaque arête correspond à une observation. La matrice d'incidence sommets-arêtes représente la base de données considérée. La classe d'une observation est représentée par la couleur de l'arête. Nous ne considérons que des problèmes à deux classes et donc deux couleurs d'arêtes. Dans la suite, nous supposerons que la coloration est non triviale, chaque couleur est présente au moins une fois.

Nous considérons l'hypergraphe $H = (V, E)$ et une fonction de coloration des arêtes $c : E \to \{0, 1\}$. Le problème d'identification de la couleur des arêtes de l'hypergraphe consiste donc à trouver un sous-ensemble C de sommets qui permet de déterminer la couleur d'une arête en connaissant son incidence à ce sous-ensemble C. Plus formellement, on cherche un sous-ensemble $C \subseteq V$ tel que pour toute paire d'arêtes $e_1 \in E$ et $e_2 \in E$ vérifiant $c(e_1) \neq c(e_2)$, on ait $e_1 \cap C \neq e_2 \cap C$.

En reprenant la littérature des codes identifiants, nous appellerons **code** un tel sous-ensemble de sommets. On remarque que les attributs correspondant aux sommets d'un code forment un support.

Le problème d'existence d'un code ne se pose pas lorsque l'hypergraphe n'a pas d'arête multiple. En effet l'ensemble des sommets est toujours un code qui permet de séparer la couleur des arêtes. Le problème d'optimisation consistant à trouver le code minimum est NP-difficile puisqu'il est une généralisation du problème de *support set* étudié dans [25].

Nous cherchons dans cette section des bornes supérieures et inférieures sur la taille d'un code permettant de distinguer la couleur des arêtes en fonction de propriétés sur la fonction de coloration et de propriétés sur la structure de l'hypergraphe. Nous allons en particulier considérer le cas extrême de l'hypergraphe complet.

5.2.2 Cas général

Le problème le plus général consiste simplement en un hypergraphe complet et une bicoloration quelconque de ses arêtes. Nous montrons que, sans restriction sur la coloration, la cardinalité d'un code minimum peut varier entre des bornes très larges. Nous donnons deux colorations des arêtes pour lesquelles la cardinalité du code atteint sa borne inférieure et supérieure. Dans le reste de ce chapitre, nous dirons qu'une borne est serrée si il existe un nombre infini de cas où cette borne est vérifiée à l'égalité.

Proposition 5.1
Soit $H = (V, E)$ un hypergraphe complet muni d'une bicoloration c de ses arêtes. La taille d'un code minimum C^* permettant de distinguer les couleurs des arêtes de H vérifie : $|C^*| \geq 1$ et cette borne est serrée.

Preuve. Soient $H = (V, E)$ un hypergraphe complet, u un sommet de V et c la fonction de coloration qui associe la couleur 1 à une arête incidente à u et la couleur 0 sinon. Le singleton $\{u\}$ est un code minimum permettant de distinguer la couleur des arêtes de H. □

Proposition 5.2
Soit $H = (V, E)$ un hypergraphe complet muni d'une bicoloration c de ses arêtes. La taille d'un code minimum C^* permettant de distinguer les couleurs des arêtes de H vérifie : $|C^*| \leq |V|$ et cette borne est serrée.

Preuve. Considérons la fonction de coloration c qui attribue la couleur 1 à l'arête de taille n et la couleur 0 à toutes les autres arêtes. Pour une telle coloration des arêtes, tous les sommets de V appartiennent à un code minimum. En effet, en considérant un code d'au plus $n - 1$ sommets, on ne sait toujours pas si l'arête considérée est celle qui est incidente à tous les sommets de V ou une autre et donc on ne peut pas distinguer leur couleur. □

Lorsque la fonction de coloration n'est pas contrainte, les bornes sur la cardinalité d'un code minimum sont donc très larges. On peut remarquer que dans une des deux colorations étudiées, il existe un grand déséquilibre entre le nombre d'arêtes de chaque couleur.

5.2.3 Méthodologie

Le problème consistant à identifier la couleur des arêtes de l'hypergraphe complet à partir de leur incidence avec un sous-ensemble de sommets est un problème NP-difficile. De plus, nous avons montré que sans restriction sur la coloration et la structure de l'hypergraphe, la taille d'une solution optimale évolue entre des bornes très larges. Nous proposons dans la suite de cette section d'introduire des conditions, d'abord sur la coloration, puis sur la structure du graphe et de regarder comment évoluent les bornes sur la taille d'une solution optimale.

Les contraintes que nous allons introduire ont pour objectif d'éliminer des configurations qui seraient aberrantes en pratique. D'autre part, ces contraintes doivent permettre de resserrer l'écart entre la borne supérieure et la borne inférieure. Avec l'introduction de telles contraintes, on peut espérer mettre en place des heuristiques pour des instances qui restent pertinentes en pratique.

La première contrainte que nous introduisons dans la Section 5.2.4 permet d'éliminer les situations où le déséquilibre entre le nombre d'arêtes de chaque couleur est grand comme dans la construction permettant de serrer la borne supérieure. Dans la Section 5.2.5, nous introduisons une contrainte supplémentaire permettant de restreindre la structure de la coloration. Cette nouvelle contrainte interdit la présence de sommets incidents uniquement à des arêtes de la même couleur comme dans la construction permettant de serrer la borne inférieure. Enfin dans la Section 5.2.6 nous nous intéressons à une structure d'hypergraphes particuliers : les graphes complets.

5.2.4 Équilibre global

Une coloration c des arêtes d'un hypergraphe $H = (V, E)$ complet est **globalement équilibrée** si le nombre d'arêtes de couleur 1 et le nombres d'arêtes de couleur 0 diffèrent d'exactement un (nous considérons que l'arête vide ne fait pas partie de l'hypergraphe, le nombre total d'arête est alors impair). Cette propriété permet d'interdire la coloration problématique pour la borne supérieure.

Pour la borne inférieure, nous pouvons observer que la construction précédente est toujours valide. En effet, il y a 2^{n-1} arêtes de la couleur 1 et $2^{n-1} - 1$ arêtes de la couleur 0, avec $n = |V|$. Malheureusement cette condition d'équilibre ne permet pas non plus de descendre la borne supérieure sur la taille d'un code.

Proposition 5.3
Soit $H = (V, E)$ un hypergraphe complet muni d'une bicoloration c de ses arêtes globalement équilibrée. La taille d'un code minimum C^* permettant de distinguer les couleurs des arêtes de H vérifie : $1 \leq |C^*| \leq |V|$ et ces bornes sont serrées.

Preuve. Considérons la coloration globalement équilibrée suivante de l'hypergraphe complet à n sommets. L'arête de taille n et les arêtes de taille 1 sont de couleur 1. Les arêtes de taille $n - 1$ sont coloriées avec la couleur 0. Les arêtes restantes ont une coloration quelconque telle que la moitié soit de couleur 0 et l'autre moitié de couleur 1. La coloration obtenue est donc globalement équilibrée. Pour distinguer la couleur de l'arête de taille n de la couleur des arêtes de taille $n - 1$, il faut interroger les n sommets du graphe. □

On se retrouve alors dans la configuration précédente avec des bornes encore très larges. La condition d'équilibre global n'est pas suffisante pour resserrer les bornes. Nous allons maintenant introduire une condition de coloration locale, pour introduire plus de structure dans une solution.

5.2.5 Équilibre local

Une coloration c des arêtes d'un hypergraphe $H = (V, E)$ complet avec au moins deux sommets est **localement équilibrée** si pour tout sommet $v \in V$, le nombre d'arêtes incidentes à v de couleur 1 est égal au nombre d'arêtes incidentes à v de couleur 0. Cette condition n'implique pas la condition d'équilibre global, mais nous allons l'utiliser comme une condition supplémentaire à l'équilibre global.

Proposition 5.4
Soit $H = (V, E)$ un hypergraphe complet muni d'une coloration c de ses arêtes localement et globalement équilibrée. La taille d'un code minimum C^* permettant de distinguer les couleurs des arêtes de H vérifie : $2 \leq |C^*| \leq |V|$ et ces bornes sont serrées.

Preuve. Nous construisons de manière récursive un hypergraphe complet à n sommets et une coloration localement et globalement équilibrée de ses arêtes. Soit $H_2 = (V_2, E_2)$ l'hypergraphe complet à deux sommets (notés v_1 et v_2) dont les arêtes de taille 1 sont de couleur 1 et l'arête de taille 2 est de la couleur 0. Cet hypergraphe est illustré Figure 5.1. Pour simplifier la récurrence, nous ajouterons à l'hypergraphe une arête vide fictive de couleur 0 que nous retirerons à la fin. On peut vérifier que l'hypergraphe H_2 est localement et globalement équilibré.

Supposons que H_k, l'hypergraphe complet à k sommets soit muni d'une coloration localement et globalement équilibrée de ses arêtes. Ajoutons le sommet v_{k+1} et toutes les arêtes manquantes pour former H_{k+1}, l'hypergraphe complet à $k+1$ sommets. Les arêtes qui étaient dans H_k conservent leur couleur. Soit e une arête sans couleur et notons $e' = e \setminus \{v_{k+1}\}$ la restriction de e à H_k. L'arête e' appartient à H_k et possède une couleur. L'arête e prend la même couleur que l'arête e'. Remarquons qu'il existe une bijection entre les arêtes incidentes à v_{k+1} et les arêtes de H_k.

La coloration de H_{k+1} ainsi obtenue est globalement équilibrée. Pour s'en convaincre, il suffit de remarquer que H_k est globalement équilibré et qu'il existe une bijection entre les nouvelles arêtes de H_{k+1} et les arêtes de H_k. La couleur de chaque nouvelle arête de H_{k+1} est celle de l'arête correspondante dans H_k. Enfin la couleur de l'arête $\{v_{k+1}\}$ permet d'obtenir l'équilibre global.

La coloration de H_{k+1} obtenue est localement équilibrée. Chaque arête incidente à v_{k+1} correspond à une unique arête de H_k (sa restriction) dont elle reprend la couleur et cette correspondance est une bijection. Puisque H_k est globalement équilibré, il y a autant d'arêtes de couleur 0 que de couleur 1 incidentes à v_{k+1}. Considérons $v \neq v_{k+1}$ un autre sommet de H_{k+1}. Ce sommet est incident à autant d'arêtes de couleur 1 que de couleur 0 dans H_k. Pour chaque nouvelle arête de E_{k+1} incidente à v, sa restriction à H_k

est incidente à v et possède la même couleur. De plus, les nouvelles arêtes de E_{k+1} sont en bijection avec les arêtes de H_k. Dans H_{k+1}, le sommet v est donc incident à autant d'arêtes de la couleur 0 que de la couleur 1.

La coloration ainsi construite permet de montrer l'existence d'une coloration localement et globalement équilibrée des arêtes d'un hypergraphe complet. D'autre part, les deux sommets v_1 et v_2 constituant H_2 forment un code permettant de connaître la couleur d'une arête de H_k. En effet, si cette arête est incidente uniquement à v_1 ou uniquement à v_2, alors elle est de couleur 1, sinon de couleur 0. D'autre part, un code minimum contient au moins deux sommets à cause de la condition d'équilibre local.

Une autre coloration peut être obtenue à partir de l'hypergraphe H_2 en coloriant une nouvelle arête e de H_{k+1} par la couleur complémentaire de l'arête e' dans H_k. Par les mêmes arguments, cette coloration est localement et globalement équilibrée. D'autre part dans cette coloration, les arêtes de taille k sont de couleur k modulo 2. On retrouve la coloration pour laquelle il faut interroger les n sommets pour connaître la couleur de l'arête. □

Figure 5.1 – L'hypergraphe H_2 avec ses arêtes de taille 1 de couleur 1 (trait plein) et son arête de taille 2 de couleur 0 (trait pointillé)

Les restrictions qui ont été imposées sur la coloration des arêtes de l'hypergraphe complet ne permettent pas d'obtenir de bornes satisfaisantes sur la cardinalité d'un code minimum. Nous allons maintenant imposer des restrictions sur la structure de l'hypergraphe. Ces restrictions empêchent les configurations qui forcent les codes à être de grande taille.

Il existe plusieurs familles d'hypergraphes pour lesquelles les arêtes ne contiennent pas d'autres arêtes. C'est le cas des hypergraphes uniformes où toutes les arêtes ont la même cardinalité. Nous allons nous concentrer sur les hypergraphes uniformes les plus simples : les graphes.

5.2.6 Graphes

Nous considérons maintenant un cas très particulier d'hypergraphe où toutes les arêtes sont de taille 2. Cette structure s'éloigne du problème pratique mais a un intérêt théorique puisqu'elle élimine les problèmes d'inclusion présents dans les hypergraphes. Nous considérons dans la suite que $H = (V, E)$ est un graphe dont les arêtes sont munies d'une bicoloration c. Le problème consiste donc à trouver un sous-ensemble de sommets $C \subseteq V$ tel que pour toute paire d'arêtes $e_1 \in E, e_2 \in E$ vérifiant $c(e_1) \neq c(e_2)$, on ait $e_1 \cap C \neq e_2 \cap C$.

Nous allons d'abord traiter, d'un point de vue algorithmique, le problème qui consiste à trouver un code minimum dans les graphes. Dans un second temps, nous allons donner des bornes sur la cardinalité d'un code minimum d'un graphe complet muni d'une bicoloration des arêtes. Enfin nous caractérisons la structure d'un code minimum.

L'ensemble des sommets V forme toujours un code permettant de distinguer les couleurs des arêtes du graphe. Le problème d'optimisation qui cherche le code de cardinalité minimum reste NP-difficile lorsque H est un graphe. Une réduction simple depuis le problème de transversal minimum (problème GT1 dans [35]) peut être faite.

Transversal minimum
Instance : Un graphe $G = (V, E)$ et un entier k
Question : Existe-t-il un ensemble S d'au plus k sommets tel que chaque arête de E soit incidente à au moins un élément de S ?

La réduction consiste à dupliquer le graphe G de l'instance de Transversal minimum en deux graphes G_0 et G_1. Les arêtes de G_0 sont coloriées avec la couleur 0 et les arêtes de G_1 avec la couleur 1. Un code minimum permettant de distinguer les couleurs de $G' = G_0 \cup G_1$ est un transversal minimum de G.

Nous considérons maintenant le graphe complet à n sommets K_n avec une bicoloration c de ses arêtes. Nous donnons maintenant des bornes générales sur cardinalité d'un code distinguant les couleurs des arêtes de K_n.

Proposition 5.5
Soit K_n le graphe complet à $n \geq 4$ sommets muni d'une coloration quelconque c de ses arêtes en deux couleurs. La taille d'un code minimum C^* permettant de distinguer les couleurs des arêtes vérifie : $1 \leq |C^*| \leq n-1$ et ces bornes sont serrées.

Preuve. La borne inférieure est serrée en considérant le graphe K_n et un sommet v tel que toutes les arêtes incidentes à v soient de couleur 1 et les autres arêtes soient de couleur 0. Le code constitué uniquement du sommet v est un code distinguant les couleurs des arêtes.

Considérons maintenant une coloration quelconque des arêtes de K_n et C un sous-ensemble de V de cardinalité $n-1$. Notons v l'unique sommet de $V \setminus C$ et considérons le code constitué des sommets de C. Les couleurs des arêtes du graphe induit par C sont parfaitement identifiées puisque chaque arête est incidente à une unique paire de sommets de C. Les arêtes manquantes sont toutes incidentes à v et intersectent chacune C en un unique sommet. On peut encore distinguer leur couleur avec l'intersection dans C.

Cette borne supérieure sur la cardinalité de C est serrée dès que le

nombre de sommets est supérieur à 5. Pour s'en convaincre considérons le graphe complet K_n dont les arêtes sont munies d'une bicoloration telle que le graphe induit par les arêtes de couleur 1 soit un cycle de taille n. Une telle coloration de K_5 est illustrée Figure 5.3.

Puisque C et $V \setminus C$ sont nécessairement non vides, il existe au moins une arête de couleur 1 entre C et $V \setminus C$. Soit (u, x) cette arête. Sans perte de généralité nous supposerons que u appartient à C et x appartient à $V \setminus C$. Si $V \setminus C$ contient au moins trois sommets alors il existe au moins une arête de couleur 0 incidente à u et à un sommet de $V \setminus C$. Il n'est donc pas possible de distinguer la couleur de cette arête de la couleur de l'arête (u, x) avec le code C. On en déduit que $V \setminus C$ contient au plus deux sommets.

Supposons maintenant que $V \setminus C$ contienne exactement deux sommets x et y. Puisque le graphe contient au moins cinq sommets, il existe, dans le cycle formé par les arêtes de couleur 1, une chaîne avec au moins quatre sommets et avec pour extrémités x et y. Notons v le voisin de u différent de x sur cette chaîne. Dans K_n, l'arête (x, u) est de couleur 1 alors que l'arête (y, u) est de couleur 0. Ces deux arêtes intersectent C en u et donc le code ne permet pas de distinguer la couleur de ces deux arêtes.

On peut aussi remarquer que la borne est serrée pour K_4 comme le montre la coloration proposée Figure 5.2. Enfin, pour K_3, on peut montrer simplement que pour toutes les colorations il existe toujours un code de taille 1. □

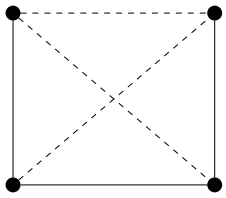

Figure 5.2 – Coloration de K_4 avec $|C^*| = 3$. Les arêtes de couleur 1 sont en trait plein, celles de couleur 0 en trait pointillé.

Nous considérons maintenant les contraintes d'équilibre local proposées dans le cadre des hypergraphes. La coloration d'un graphe est localement équilibrée si pour chaque sommet u, le nombre d'arêtes de couleur 1 incidentes à u et le nombre d'arêtes de couleur 0 incidentes à u diffèrent d'au plus un. Cette contrainte permet d'interdire la coloration proposée dans la démonstration de la proposition précédente à partir de sept sommets.

Proposition 5.6
Soit K_n le graphe complet à n sommets muni d'une coloration localement équilibrée c de ses arêtes en deux couleurs. La taille d'un code minimum C^* permettant de distinguer les couleurs des arêtes vérifie : $\lfloor \frac{n-1}{2} \rfloor + 1 \leq |C^*|$ et cette bornes est serrée.

Preuve. Considérons un sommet de K_n. Puisque la coloration des arêtes est localement équilibrée, la moitié des arêtes incidentes à u sont de la couleur 0, l'autre moitié de la couleur 1. Pour distinguer la couleur de ces arêtes, il faut au moins $\lfloor \frac{n-1}{2} \rfloor$ sommets supplémentaires dans le code. On en déduit l'inégalité suivante : $|C^*| \geq \lfloor \frac{n-1}{2} \rfloor + 1$.

Il est possible de serrer cette borne. On considère le graphe K_{2p} dont les arêtes sont coloriées de la manière suivante. Les sommets sont partitionnés en deux classes : V_1 et V_2, chacune de cardinalité p. Les arêtes du graphe induit par V_1 et celles du graphe induit par V_2 sont coloriées avec la couleur 1 alors que celles entre V_1 et V_2 sont coloriées avec la couleur 0. Le graphe ainsi formé est localement équilibré puisque chaque sommet est incident à $p-1$ arêtes de couleur 1 et p arêtes de couleur 0. L'ensemble V_1 forme un code distinguant les couleurs des arêtes : les arêtes qui sont incidentes à zéro ou deux sommets de V_1 sont de la couleur 1, les autres de la couleur 0. Ce code serre l'inégalité sur la cardinalité d'un code minimum. □

La construction proposée dans la preuve précédente permet de serrer la borne supérieure sur la taille d'un code minimum uniquement lorsque le nombre de sommets est pair. Lorsque le nombre de sommets est impair, trouver une coloration localement équilibrée n'est pas toujours possible. En effet, il est impossible de trouver une coloration localement équilibrée d'un graphe avec un nombre impair de sommets et un nombre impair d'arêtes. Supposons qu'une telle coloration existe alors chaque sommet est incident à un nombre pair d'arêtes. De plus, chaque sommet est incident à autant d'arêtes de couleur 0 que de couleur 1. Il y a donc autant d'arêtes de couleur 1 que d'arêtes de couleur 0 dans le graphe. Mais le nombre total d'arêtes est impair ce qui nous amène à une contradiction. Ce cas se produit dans le graphe complet lorsque le nombre de sommets est congru à 3 modulo 4.

Proposition 5.7
Soit K_n le graphe complet à n sommets muni d'une coloration localement équilibrée c de ses arêtes en deux couleurs. La taille d'un code minimum C^* permettant de distinguer les couleurs des arêtes vérifie : $|C^*| \leq n-1$ et cette borne est serrée.

Preuve. Pour montrer que la borne est serrée, nous allons donner une coloration localement équilibrée du graphe K_{5^d} pour tout entier d supérieur ou égal à 1.

Soit K_5 le graphe complet à cinq sommets dont la coloration est donnée Figure 5.3. Ce graphe est localement équilibré et le code minimum vérifie $|C^*| = n-1$ comme le montre la preuve de la Proposition 5.5.

La coloration de $K_{5^{d+1}}$ se fait à partir de la coloration de K_{5^d} en remplaçant chaque sommet i de K_5 par une copie coloriée de K_{5^d} étiquetée i. Les arêtes entre la copie étiquetée i et la copie étiquetée j sont de la couleur

de l'arête (i, j) dans K_5. On peut montrer que la coloration de ce graphe reste localement équilibrée.

Nous allons montrer par induction qu'un code minimum de K_{5^d} munie de la coloration définie précédemment vérifie $|C^*| = n - 1$. Supposons que la coloration K_{5^d} vérifie $|C^*| = n - 1$ et supposons qu'il existe un code C de $K_{5^{d+1}}$, colorié par la construction, tel que $|C| < n - 1$. Considérons u et v deux sommets hors du code.

Si u et v sont dans la même copie de K_{5^d} d'étiquette i, considérons le graphe induit par la copie i. Puisque C est un code, dans le graphe induit par la copie i, les sommets dans le code permettent de distinguer les couleurs des arêtes. D'après l'hypothèse d'induction, la coloration de K_{5^d} vérifie $|C^*| = n - 1$ et donc u et v ne peuvent appartenir à la même copie.

Soient i et j $(i \neq j)$ les étiquettes des copies de K_{5^d} contenant respectivement les sommets u et v. En prenant un sommet quelconque dans chacune des trois copies restantes, on obtient comme graphe induit le graphe complet à cinq sommets qui, par construction, possède la coloration définie pour K_5 dans la Figure 5.3. Dans ce graphe un code nécessite au moins quatre sommets pour identifier les couleurs des arêtes. □

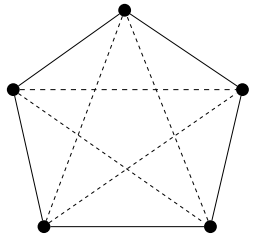

Figure 5.3 – Coloration de K_5 localement équilibrée. Les arêtes de couleur 1 sont en trait plein, celles de couleur 0 en trait pointillé.

Le graphe donné comme exemple pour serrer la borne inférieure sur la cardinalité de C donne une idée de la structure d'un code. Nous pouvons montrer que tous les codes distinguant les couleurs des arêtes ont la même structure. Les sommets sont partitionnés en deux ensembles : C et $V \setminus C$ et les arêtes en trois ensembles E_1, E_2 et E_3. Les arêtes de E_1 ont leurs deux extrémités dans C, les arêtes de E_2 ont leurs deux extrémités dans $V \setminus C$ et les arêtes de E_3 ont une extrémité dans chaque ensemble. Le graphe $G_1 = (C, E_1)$ peut avoir une coloration quelconque de ses arêtes, puisque chaque arête est incidente à une unique paire de sommets de C. Le graphe $G_2 = (V \setminus C, E_2)$ a toutes ses arêtes de la même couleur puisqu'elles n'intersectent aucun sommet du code. Enfin le graphe $G_3 = (V, E_3)$ est un graphe biparti tel que toutes les arêtes incidentes à un sommet de C sont de la même couleur. En effet, ces arêtes étant incidentes à un unique sommet du code,

si elles ne sont pas de la même couleur alors il n'est pas possible de les distinguer.

La restriction forte qui consiste à considérer le problème sur des graphes au lieu des hypergraphes ne rend pas le problème d'optimisation plus simple. Cependant, les bornes sur la cardinalité d'un code minimum pour le graphe complet muni d'une coloration des arêtes sont plus sensibles aux conditions d'équilibre. La restriction au cas des graphes nous permet de plus de donner des indications sur la structure d'un code.

Dans cette section nous avons abordé un problème d'identification des couleurs des arêtes d'un hypergraphe. Le problème consistant à trouver un code de cardinalité minimale est NP-difficile dans les hypergraphes. Nous avons tenté de regarder comment se comportent les solutions lorsque des contraintes sont imposées à l'instance. En particulier, nous avons défini deux notions d'équilibre pour les coloration et nous avons regardé le problème sur des graphes au lieu d'hypergraphes.

Les deux conditions d'équilibre sur la coloration des arêtes proposées pour les hypergraphes ne permettent pas d'obtenir des bornes satisfaisantes sur la taille d'un code minimum. En effet les bornes obtenues vont de 1 sommet (2 dans le cas de colorations localement équilibrées) jusqu'à l'ensemble des sommets de l'hypergraphe. Le problème étudié a tout de même l'avantage de garantir l'existence d'une solution.

En regardant plus précisément les colorations qui forcent à utiliser tous les sommets dans un code minimum, nous pouvons remarquer qu'elles conservent toujours la même structure. L'arête à n sommets est d'une couleur alors que toutes les arêtes à $n-1$ sommets sont de la couleur opposée. En pratique cela traduit une base de donnée très particulière où toutes les observations qui ont $n-1$ attributs positifs sont d'une classe et l'observation qui a l'ensemble de ses attributs positifs est de la classe opposée. De manière générale, si la coloration des arêtes n'est pas restreinte, les problèmes d'inclusions présentés précédemment créent des configurations aberrantes en pratique.

Pour éliminer ces problèmes d'inclusion d'arêtes, nous avons étudié une structure d'hypergraphe plus simple : les graphes. Malgré cette forte restriction, le problème reste NP-difficile et les codes minimum peuvent être de tailles très différentes, même en ajoutant des contraintes sur la coloration.

Pour l'instant, nous n'avons pas posé de restrictions sur la correspondance faite entre l'intersection d'une arête dans le code et sa couleur.

5.3 Sélection d'attributs binaires avec contrainte de seuil

Dans cette section, nous allons considérer un code particulier où la couleur d'une arête est identifiée en comparant la taille de son intersection avec le code et un seuil k. En pratique, cela revient à trouver un sous-ensemble d'attributs tel que la présence d'au moins k de ces attributs dans une observation indique son appartenance à la classe 1.

5.3.1 Modélisation

Soit $H = (V, E)$ un hypergraphe et $c : E \to \{0, 1\}$ une bicoloration des arêtes de H. Nous cherchons à identifier la couleur des arêtes de H avec un code particulier. Ce code est défini par une paire (S, k) où S est un sous-ensemble de sommets et k un entier. L'identification de la couleur d'une arête e se fait en comptant le nombre de sommets de S incidents à e et en comparant le résultat à l'entier k. Ainsi, une arête est identifiée comme étant de couleur 1 si et seulement si $|e \cap S| \geq k$. Nous appelons **ensemble k-distinguant** une telle paire (S, k). Dans une application médicale, cet ensemble revient à sélectionner un groupe de tests tel que la présence d'au moins k tests positifs suffit à déclarer l'observation comme appartenant à la classe 1. Par exemple avoir au moins deux symptômes parmi : "fièvre", "migraine", "courbatures" permet de diagnostiquer la grippe.

Le problème d'existence d'une telle paire (S, k) se pose lorsque l'hypergraphe H et la coloration c sont quelconques. En effet si une arête e_1 de couleur 1 est strictement incluse dans une arête e_2 de couleur 0, il n'existera pas de paire (S, k) permettant de retrouver la couleur des arêtes e_1 et e_2. Il est important de noter l'asymétrie entre la couleur 0 et la couleur 1. En particulier, une arête de couleur 0 peut être incluse dans une arête de couleur 1 sans empêcher l'identification des couleurs alors que l'inverse n'est pas possible.

Nous montrons que le problème d'existence d'un ensemble distinguant est un problème difficile au sens de la complexité (Proposition 5.9). Nous étudions ensuite le problème d'existence dans une famille particulière d'hypergraphes : les plans projectifs. Nous verrons que dans le plan de Fano l'existence d'un ensemble distinguant est garantie pour toute coloration non triviale (Théorème 5.1).

5.3.2 Aspects algorithmiques

Nous nous intéressons dans cette section à l'existence d'un ensemble distinguant d'un point de vue algorithmique. Plus spécifiquement nous allons étudier la complexité des deux problèmes suivants :

Ensemble distinguant
Instance : Un hypergraphe $H = (V, E)$ et une coloration de ses arêtes $c : E \to \{0, 1\}$.
Question : Existe-t-il un sous-ensemble de sommets S et un entier k tels que pour toute arête e, $|e \cap S| \geq k \iff c(e) = 1$?

Ensemble k-distinguant
Instance : Un hypergraphe $H = (V, E)$ et une coloration de ses arêtes $c : E \to \{0, 1\}$.
Question : Existe-t-il un sous-ensemble de sommets S tel que pour toute arête e, $|e \cap S| \geq k \iff c(e) = 1$?

Il existe une réduction triviale entre les problèmes Ensemble distinguant et Ensemble k-distinguant. En effet une solution du problème Ensemble distinguant peut être trouvée en résolvant le problème Ensemble k-distinguant pour toutes les $|V|$ valeurs possibles de k.

Proposition 5.8
Le problème Ensemble 1-distinguant est polynomial.

Preuve. Un sommet de S ne peut être incident qu'à des arêtes de la couleur 1, par définition d'un ensemble 1-distinguant. On en déduit alors qu'un sommet incident à au moins une arête de couleur 0 ne peut être présent dans S. Si un ensemble 1-distinguant S existe, alors ajouter à S un sommet incident uniquement à des arêtes de couleur 1 crée un nouvel ensemble S' lui même 1-distinguant. En répétant ce procédé, on obtient que si il existe un ensemble 1-distinguant alors l'ensemble des sommets incidents uniquement à des arêtes de couleur 1 est un ensemble 1-distinguant. Vérifier qu'un ensemble de sommets est un ensemble 1-distinguant peut se faire en temps polynomial. □

Les autres problèmes, Ensemble distinguant et Ensemble k-distinguant pour $k \geq 2$, sont difficiles.

Proposition 5.9 (Pitt et Valiant [75])
Les problèmes Ensemble distinguant et Ensemble k-distinguant ($k \geq 2$) sont NP-complets.

Ce résultat est connu dans le contexte des fonctions booléennes. Un hypergraphe H bicoloré par $\{0, 1\}$ correspond exactement à une fonction booléenne partiellement définie f. L'existence d'un ensemble distinguant pour H est équivalente à l'existence d'une fonction à seuil (définition dans [25]) avec coefficients dans $\{0, 1\}$ pour la fonction f. La complexité de l'existence d'une telle fonction à seuil a été montrée dans [75]. Nous propo-

sons ci-dessous une preuve alternative dans le formalisme des ensembles k-distinguant.

Preuve. Nous allons effectuer une réduction depuis le problème 3SAT (problème LO2 dans [35]) :

3SAT
Instance : Un ensemble U de variables et une collection C de clauses (disjonction de littéraux) sur U telle que chaque clause $c_j \in C$ vérifie $|c_j| = 3$.
Question : Existe-t-il une affectation des variables de U permettant de satisfaire toutes les clauses de C ?

À partir d'une instance de 3SAT avec C non vide, nous allons construire une instance du problème Ensemble distinguant. L'ensemble des sommets V de l'hypergraphe est constitué de :
- deux sommets v_i et $\bar{v}_i$ pour chaque variable $x_i \in U$ et son complément ;
- deux sommets u_j et u'_j pour chaque clause $c_j \in C$;
- deux sommets s et t.

Les arêtes E de l'hypergraphe ainsi que leur couleur c sont définies de la manière suivante :
- une arête $\{u_j, u'_j\}$ de couleur 0 pour chaque clause $c_j \in C$;
- une arête $\{v_i, \bar{v}_i\}$ de couleur 0 pour chaque variable $x_i \in U$;
- une arête $\{s, t\}$ de couleur 1 ;
- une arête de couleur 1 pour chaque clause $c_j \in C$ incidente à u_j et aux trois sommets correspondant aux variables, éventuellement complémentées, présentes dans la clause.

L'hypergraphe $H = (V, E)$ dont les arêtes sont coloriées par c possède une arête de couleur 1, incidente à deux sommets seulement : l'arête $\{s, t\}$. Ainsi, s'il existe une solution au problème Ensemble distinguant sur cette instance, alors le seuil k ne peut prendre que les valeurs 1 ou 2. Si C est non vide, il existe au moins une arête de couleur 1 distincte de $\{s, t\}$. Cette arête intersecte des sommets contenus dans des arêtes de couleur 0 (arêtes $\{u_j, u'_j\}$ et $\{v_i, \bar{v}_i\}$). Le seuil k ne peut alors pas prendre la valeur 1. Dans une solution de cette instance, le seuil k est nécessairement égal à 2.

Montrons maintenant qu'il existe une solution à cette instance d'Ensemble distinguant si et seulement si il existe une solution à l'instance originale de 3SAT.

Supposons qu'il existe une affectation des variables de U permettant de satisfaire chaque clause $c_j \in C$. Notons V^1 l'ensemble des sommets v_i tels que la variable x_i est affectée à la valeur "Vrai" dans la solution de 3SAT. De la même manière, notons V^2 l'ensemble des sommets $\bar{v}_i$ tels que la variable x_i est affectée à la valeur "Faux" dans la solution de 3SAT. L'ensemble S constitué des sommets de V^1, des sommets de V^2, des sommets s, t et des

sommets u_j pour chaque clause $c_j \in C$ forme une solution de ENSEMBLE DISTINGUANT avec $k = 2$. En effet chaque arête de couleur 0 intersecte au plus un seul sommet de S alors que les arêtes de couleur 1 intersectent chacune au moins deux sommets de S.

Réciproquement, supposons qu'il existe une solution (S, k) à l'instance du problème ENSEMBLE DISTINGUANT construite à partir de l'instance de 3SAT. Nous avons vu que dans une telle solution, la valeur du seuil k était nécessairement 2. Soit $S' = \{v_i \in S\} \cup \{\bar{v}_i \in S\}$. Puisque, par définition, chaque arête de couleur 0 intersecte S au plus une fois, un seul des deux sommets v_i et $\bar{v}_i$ peut appartenir à S et donc à S'. D'autre part chaque arête de taille quatre est incidente à S en au moins deux sommets puisqu'elle est de couleur 1 et donc à au moins un sommet de S'. En effet ces arêtes de taille 4 ont exactement trois sommets susceptibles d'appartenir à S'. En affectant à "Vrai" les variables x_i correspondant aux sommets v_i de S', à "Faux" les variables x_j correspondant aux sommets $\bar{v}_i$ de S' et une valeur quelconque aux variables restantes, nous obtenons une solution de l'instance de 3SAT. En effet à chaque variable est affectée une unique valeur et chaque clause est satisfaite par au moins une de ses variables.

Afin d'illustrer cette réduction, considérons l'instance suivante de 3SAT. L'ensemble de variables est $U = \{x_1, x_2, x_3, x_4\}$ et l'ensemble des clauses est $C = \{\{x_1, \bar{x}_3, \bar{x}_4\}, \{\bar{x}_1, x_2, \bar{x}_4\}\}$. L'hypergraphe $H = (V, E)$ et la coloration des arêtes c associée sont présentés Figure 5.4. Les arêtes de taille quatre incidentes aux sommets u_i sont représentées par des surfaces grisées. Les arêtes en trait plein sont de la couleur 1 alors que celles en pointillé sont de la couleur 0. □

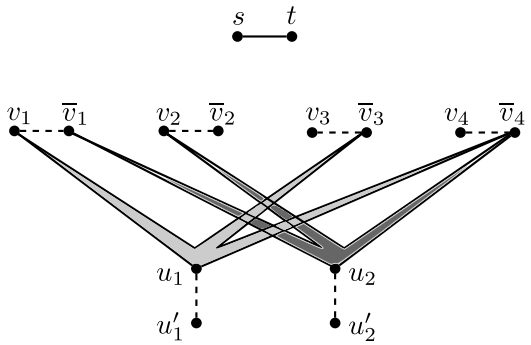

Figure 5.4 – Hypergraphe construit à partir d'une instance de 3SAT.

L'existence d'un ensemble distinguant est un problème difficile au sens de la complexité. La condition d'inclusion présentée en introduction de cette section est nécessaire mais n'est pas suffisante pour garantir l'existence d'un ensemble distinguant. Nous allons regarder ce problème dans des hypergraphes particuliers : les plans projectifs. Ces hypergraphes ont une struc-

ture très régulière et la condition d'inclusion est vérifiée. Les graphes sont aussi des hypergraphes ayant ces propriétés mais les ensembles distinguants ont des structures simples et identifiables en temps polynomial.

5.3.3 Plans projectifs

Un **plan projectif** d'ordre n est un hypergraphe sur n^2+n+1 sommets tel que :

- toute paire de sommets est contenue dans une unique arête,
- deux arêtes s'intersectent en un unique sommet,
- chaque sommet est contenu dans exactement $n+1$ arêtes,
- chaque arête contient exactement $n+1$ sommets.

On note $\mathbb{P}_n$ le plan projectif d'ordre n. On sait que $\mathbb{P}_n$ existe si n est la puissance d'un nombre premier. Le plan projectif d'ordre 2 est connu sous le nom de plan de Fano, il est représenté Figure 5.5. Pour plus d'informations sur les plans projectifs et plus généralement sur les *designs*, nous renvoyons le lecteur au Chapitre 19 de [89].

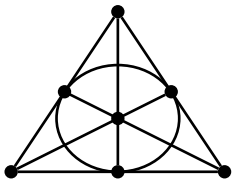

Figure 5.5 – Plan projectif d'ordre 2. Les arêtes sont les six segments plus le cercle central.

Nous allons étudier l'existence d'ensembles distinguants dans les plans projectifs et plus particulièrement dans le plan de Fano noté $\mathbb{P}_2$. Nous montrons que pour toute bicoloration des arêtes de $\mathbb{P}_2$, il existe un ensemble distinguant (S, k). Nous entendons par coloration triviale, une coloration des arêtes avec une unique couleur.

Nous commençons par montrer que dans toute coloration de $\mathbb{P}_2$, il existe un sommet contenu dans des arêtes de la même couleur (Lemme 5.1). Pour un graphe G, muni d'une bicoloration c des arêtes nous définissons la coloration complémentaire c' qui à une arête de couleur 0 dans c associe la couleur 1 dans c' et à une arête de couleur 1 dans c associe la couleur 0 dans c'. Nous montrons qu'il existe pour toute bicoloration des arêtes de $\mathbb{P}_2$ un ensemble distinguant et un ensemble distinguant de la coloration complémentaire (Théorème 5.1).

Lemme 5.1
Soit $\mathbb{P}_2$ le plan de Fano avec une coloration des arêtes en deux couleurs. Alors il existe un sommet contenu dans trois arêtes de la même couleur.

Preuve. Par l'absurde, supposons que la propriété soit fausse. Considérons alors un sommet x de $\mathbb{P}_2$. Sans perte de généralité, il est incident à deux arêtes de couleur 1 et une arête de couleur 0. Cette situation est illustrée Figure 5.6. En conservant les notations de la figure, considérons l'arête $\{d, e, f\}$.

Premier cas, l'arête $\{d, e, f\}$ est de couleur 1. Par conséquent, l'arête $\{a, b, f\}$ est de couleur 0 mais alors l'arête $\{a, c, e\}$ ne peut être coloriée sans créer un sommet contenu dans trois arêtes de la même couleur.

Deuxième cas, l'arête $\{d, e, f\}$ est de couleur 0. Par conséquent, l'arête $\{d, b, c\}$ est de couleur 1, l'arête $\{a, c, e\}$ est forcée à être de couleur 0 et il est impossible de donner une couleur à $\{a, b, f\}$ sans se retrouver avec un sommet contenu dans trois arêtes de la même couleur.

Il n'est donc pas possible de bicolorier les arêtes de $\mathbb{P}_2$ sans créer de sommet contenu uniquement dans des arêtes de la même couleur. □

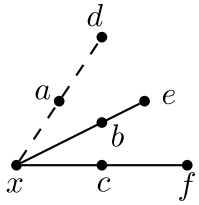

Figure 5.6 – Illustration du voisinage du sommet x. Les couleurs des arêtes sont représentées par des traits pleins (couleur 1) ou pointillés (couleur 0). Les arêtes manquantes sont pour l'instant sans couleur.

Nous nous intéressons maintenant aux bicolorations non triviales des arêtes de $\mathbb{P}_2$. Pour rappel, une coloration est triviale si toutes les arêtes sont de la même couleur. Dans ce cas le problème Ensemble distinguant admet une solution triviale.

Théorème 5.1
Soit $\mathbb{P}_2$ le plan de Fano avec une bicoloration des arêtes. Alors il existe un ensemble distinguant pour cette coloration.

Preuve. En appliquant le Lemme 5.1, nous savons qu'il existe un sommet x contenu dans des arêtes de la même couleur. Nous supposerons que x est contenu dans des arêtes de couleur 1. En raisonnant sur le nombre d'arêtes de couleur 1 dans cette configuration, nous montrons qu'il existe à chaque fois un ensemble distinguant pour la coloration et un ensemble distinguant pour la coloration complémentaire.

Cas 1 : $\mathbb{P}_2$ contient exactement trois arêtes de couleur 1. Par hypothèse, ces arêtes sont incidentes au sommet x. L'ensemble $\{x\}$ est donc un ensemble

1-distinguant et l'ensemble $V \setminus \{x\}$ est un ensemble 3-distinguant pour la coloration complémentaire.

Cas 2 : $\mathbb{P}_2$ contient exactement quatre arêtes de couleur 1. Par hypothèse, trois de ces arêtes sont incidentes à x. Sans perte de généralité, on suppose que l'arête $\{d, e, f\}$ est la dernière arête avec la couleur 1 (voir Figure 5.7). L'ensemble $S_1 = \{x, d, e, f\}$ est un ensemble 2-distinguant pour la couleur 1. En effet, d'après les propriétés des plans projectifs, deux sommets appartiennent à une unique arête. Parmi les six couples de sommets de S_1, les couples $\{d, e\}$, $\{d, f\}$, $\{e, f\}$ appartiennent à la même arête $\{d, e, f\}$. Les couples restants appartiennent à trois arêtes distinctes, adjacentes à x et par définition de couleur 1. De même, il est facile de voir que l'ensemble $S_0 = V \setminus S_1$ est un ensemble 2-distinguant pour la coloration complémentaire.

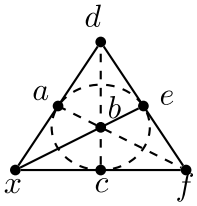

Figure 5.7 – Illustration du Cas 2. Les arêtes de couleur 1 sont en trait plein, celles de couleur 0 en trait pointillé.

Cas 3 : $\mathbb{P}_2$ contient exactement cinq arêtes de couleur 1 et deux arêtes de couleur 0. Par définition ces deux arêtes s'intersectent en un sommet. Sans perte de généralité notons c ce sommet. Cette situation est illustrée Figure 5.8. L'ensemble $S_1 = \{x, f\}$ forme un ensemble 1-distinguant. L'ensemble $S_0 = \{a, b, c, d, e\}$ est un ensemble 3-distinguant pour la coloration complémentaire.

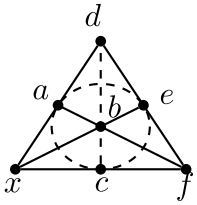

Figure 5.8 – Illustration du Cas 3. Les arêtes de couleur 1 sont en trait plein, celles de couleur 0 en trait pointillé.

Cas 4 : $\mathbb{P}_2$ contient exactement six arêtes de couleur 1. Sans perte de généralité soit $\{b, c, d\}$ l'unique arête de couleur 0. L'ensemble $S_1 = V \setminus \{b, c, d\}$ n'est contenu que dans des arêtes de couleur 1, c'est un ensemble 1-distinguant. Clairement l'ensemble $S_0 = \{b, c, d\}$ est un ensemble 3-distinguant pour la coloration complémentaire. □

Le plan de Fano est une structure intéressante pour le problème d'Ensemble distinguant puisque pour toute bicoloration de ses arêtes, le problème admet une solution. Malheureusement ce résultat ne s'étend pas aux plans de dimension supérieure.

Proposition 5.10
Il existe une bicoloration des arêtes de $\mathbb{P}_3$ telle que le problème Ensemble distinguant sur cette instance n'admette pas de solution.

Idée de preuve. Soit x un sommet de $\mathbb{P}_3$, il appartient à exactement quatre arêtes. Colorions ces arêtes avec la couleur 1 et considérons un autre sommet y. Parmi les arêtes qui lui sont incidentes, une est déjà de la couleur 1. Colorions deux autres arêtes incidentes à y avec la couleur 1. Les sept arêtes restantes sont coloriées avec la couleur 0. En traitant les quatre valeurs de seuil possibles, on peut montrer par "forçage" qu'il n'est pas possible d'avoir un ensemble distinguant pour cette instance.

L'identification de la couleur d'une arête par un ensemble distinguant est un cas particulier du problème d'identification étudié dans la section précédente. Dans cette variante, l'existence d'un ensemble distinguant n'est pas garantie pour toutes les colorations. Des conditions nécessaires sur l'inclusion des arêtes de différentes couleurs apparaissent naturellement. Il nous semble plus difficile de donner une caractérisation simple de l'existence d'un ensemble distinguant. D'un point de vue algorithmique, nous avons montré que ce problème est NP-complet dans le cas général en donnant une réduction depuis le problème 3SAT.

Nous nous sommes ensuite intéressés à des instances particulières où l'hypergraphe considéré est un plan projectif. L'intérêt des plans projectifs est de fournir un hypergraphe régulier, avec un nombre d'arêtes de l'ordre du nombre de sommets et dont les intersections sont uniques. Nous avons montré que l'existence d'un tel code est garantie dans le plan de Fano mais que ces résultats ne peuvent pas être généralisés à tous les plans projectifs.

5.4 Conclusion

Le problème de sélection d'attributs a des intérêts pratiques en analyse de données. Il permet de réduire la taille des instances des algorithmes d'apprentissage en supprimant des redondances et des attributs non pertinents. Un bon sous-ensemble d'attributs permet de lutter contre les phénomènes de sur-apprentissage. Les approches classiques de la littérature consistent à évaluer les sous-ensembles d'attributs par des critères extérieurs (*filters*) ou par les performances d'un algorithme d'apprentissage sur ce sous-ensemble

(*wrappers*). Le problème a aussi été traité dans la littérature de manière combinatoire dans le cadre de l'Analyse Combinatoire de Données. Ces problèmes combinatoires peuvent se voir comme une généralisation des problèmes de couverture par des tests de la littérature. Nous avons proposé une première étude du problème de sélection d'attributs dans le cas de données binaires.

Nous avons proposé une modélisation de la sélection d'attributs binaires comme un problème d'identification de couleur des arêtes d'un hypergraphe par un sous-ensemble minimum de sommets. Ce problème d'optimisation étant NP-difficile, nous avons adopté une approche structurelle. L'objectif étant de trouver des restrictions sur la coloration, la structure de l'hypergraphe ou sur la fonction permettant l'identification. Dans le cas de l'hypergraphe complet, nous avons montré qu'un code minimum pouvait être constitué d'un sommet ou de l'ensemble des sommets en fonction de la coloration des arêtes. Des conditions simples sur la coloration telles que l'équilibre local ou global ne sont pas suffisantes pour avoir des bornes plus serrées sur la cardinalité d'un code minimum.

En restreignant la structure de l'hypergraphe à un graphe, le problème d'identification reste difficile. Dans les graphes complets, un code minimum peut être constitué d'un sommet comme de l'ensemble des sommets moins un. En reprenant les restrictions sur la coloration des hypergraphes, les bornes se resserrent mais l'écart entre borne supérieure et borne inférieure reste de l'ordre du nombre de sommets. Le principal apport de cette restriction aux graphes est de permettre une description simple de la structure d'un code minimum.

La structure des graphes est trop restrictive et s'éloigne du problème original. Nous avons regardé une variante du code minimum pour laquelle l'identification d'une arête se fait avec la cardinalité de son intersection avec le code. L'existence d'un tel code n'est pas assurée et la déterminer est un problème NP-complet. Nous avons montré sur le plan de Fano, qu'il était possible de garantir l'existence d'un tel code pour toute coloration de ses arêtes. Ces résultats ne s'étendent malheureusement pas à tous les plans projectifs.

Le problème de sélection d'attributs par des méthodes combinatoires est un problème difficile. Il est en effet difficile de trouver une structure combinatoire représentant bien le problème pratique. Nous avons évoqué le problème d'inclusion des arêtes qui nous éloigne du problème réel et fait augmenter la taille du code. D'autre part, l'asymétrie entre couleur 0 et couleur 1 dans le cas des ensembles distinguants nécessite des structures pertinentes en pratique où leur existence est garantie.

Il reste possible de résoudre le problème de sélection d'attributs en pratique, de manière exacte. Par exemple, en utilisant les algorithmes de résolutions de programmes linéaires en nombres entiers tels que le *branch & bound* si la taille de données est raisonnable. L'avantage de ces méthodes par

rapport aux méthodes classiques est de fournir une garantie d'optimalité. En revanche, ces méthodes sont plus sensibles aux bruits dans les données et à la taille des instances. Au-delà des aspects pratiques, une grande variété de problèmes théoriques se cache derrière la sélection d'attributs binaires.

Conclusion

Le travail de thèse présenté dans ce manuscrit traite de l'exploration de données avec le point de vue de la recherche opérationnelle. Les deux domaines font face à des problèmes de grande taille et algorithmiquement difficiles. On peut alors s'intéresser à la mise en place d'heuristiques ainsi qu'à l'étude de leur complexité. Cette thèse se focalise sur les problèmes de classification supervisée et plus particulièrement sur la méthode de l'Analyse Combinatoire de Données (ACD). Nous proposons pour cette méthode de nouveaux algorithmes, une application médicale originale ainsi qu'une nouvelle extension à l'analyse de temps de survie. Un autre aspect de cette thèse concerne l'étude de problèmes combinatoires issus de l'exploration de données.

Le Chapitre 1 est une introduction à l'exploration de données. Il donne les définitions classiques ainsi que les étapes du processus d'apprentissage : préparation des données, algorithme d'apprentissage, évaluation du modèle généré. Les premiers liens avec la recherche opérationnelle sont présentés au travers d'une méthode de classification reposant sur la programmation linéaire. Ce premier chapitre présente ensuite trois méthodes classiques de la littérature : les machines à vecteur de support, les arbres de décision et les *k-means*. La dernière partie traite des liens entre exploration de données et recherche opérationnelle. En premier lieu, nous présentons comment les problèmes d'exploration de données sont modélisés en problèmes d'optimisation avec un objectif partiellement connu. Nous donnons ensuite quelques applications d'heuristiques d'exploration de données pour résoudre des problèmes de recherche opérationnelle.

Les méthodes présentées dans ce chapitre utilisent des modèles simples pour éviter le sur-apprentissage. Les résultats théoriques de Vapnik suggèrent que de bons modèles peuvent être générés en cherchant un compromis entre performances empiriques et complexité du modèle. Les résultats issus de la théorie de Vapnik donnent une formulation de la complexité du modèle. La résolution de ces problèmes multiobjectifs est une problématique s'inscrivant dans le domaine de la recherche opérationnelle et contribuant au développement de l'analyse de données.

Le Chapitre 2 présente l'analyse combinatoire de données, une méthode

de classification supervisée reposant sur la théorie des fonctions booléennes et leurs extensions. Le chapitre présente d'abord le schéma général de l'ACD avec une revue des méthodes de la littérature.

Dans un second temps, un nouvel algorithme de génération de motifs reposant sur les fonctions booléennes duales et l'échantillonnage des données est décrit. Cet algorithme génère des modèles à vaste marge à partir de motifs fortement homogènes. Les résultats théoriques et pratiques indiquent que cette méthode génère des modèles performants mais reste sensible au bruit des données. Les motifs générés avec un faible échantillon d'observations sont de bonne qualité (homogénéité et prévalence). La sélection du modèle à partir des motifs générés est faite avec un simple algorithme glouton. Un algorithme plus évolué, basé sur un programme linéaire en nombres entiers, peut fournir des modèles avec plus de motifs et moins de conflits.

Ce chapitre se termine par une application concrète de l'ACD sur des données médicales en partenariat avec l'hôpital Avicenne de Paris. Les modèles sont générés à partir de méthodes de la littérature adaptées aux contraintes médicales. Les modèles générés sont meilleurs que d'autres modèles à base de règles tels que les arbres de décisions. Les résultats de cette étude feront l'objet d'une publication dans le domaine de la recherche médicale.

Le Chapitre 3 présente une extension de l'ACD pour traiter des problèmes d'analyse de temps de survie. Dans ces problèmes, l'objectif est de prédire une valeur numérique à partir d'observations éventuellement censurées. Pour ces observations, la valeur n'est pas connue mais on dispose d'une borne inférieure sur celle-ci (par exemple lorsque la fin de l'étude arrive sans que l'événement étudié ne se soit produit). La méthode proposée se base sur les techniques classiques de l'ACD : génération de motifs puis sélection d'un modèle permettant d'expliquer le maximum d'observations. La génération de motifs est exhaustive, mais les motifs sont regroupés par familles de motifs similaires. Ces familles permettent de donner différentes caractérisations d'un groupe d'observations ayant des propriétés de survie similaires. Les performances des modèles générés sont du même ordre de grandeur que les modèles de la littérature tout en restant interprétables.

L'algorithme de génération exhaustif de motifs peut être amélioré en prenant en compte les observations couvertes par les motifs et éviter de générer trop de motifs synonymes. Un tel algorithme pourrait être appliqué aux données contenant un grand nombre d'attributs et pour lesquelles il n'est pas possible de faire une énumération exhaustive des motifs même avec un faible degré.

L'application de ces algorithmes sur des données réelles avec des chercheur en médecine permettrait d'améliorer la méthode. En particulier, la construction du graphe de similarités pourrait prendre en compte des contraintes liées à la recherche médicale. Enfin, une mise en situation permettrait d'avoir une évaluation par des experts des modèles générés en plus des

performances. Une étude a été initiée avec l'hopital de Lyon sur l'analyse du risque de récidive du lymphome malin.

La Chapitre 4 traite d'un problème de partition de graphe en graphes denses. Ce problème est apparu lors de l'étude du graphe de similarités pour l'analyse de temps de survie. C'est un problème issu de l'exploration de données et plus spécifiquement de l'apprentissage non supervisé. L'objectif est de maximiser la somme des degrés moyens de chaque classe. Nous montrons que ce problème est NP-difficile en général et nous donnons un algorithme polynomial pour les arbres. Cet algorithme repose sur le lien fort qui existe entre la partition optimale dans les arbres et le couplage maximum.

Une perspective immédiate serait d'étudier la complexité du problème dans les graphes bipartis. En effet, certains résultats de structures des solutions optimales s'étendent aux graphes bipartis, mais pas la structure de partition en étoiles.

À plus long terme, une étude comparative des différentes heuristiques proposées dans le domaine de la détection de communautés permettrait de guider le choix d'une heuristique en fonction de l'application. Il serait intéressant de comparer les solutions fournies par les heuristiques classiques de la littérature aux solutions optimales.

Le Chapitre 5 propose différentes modélisations du problème de sélection d'attributs binaires. Ce problème généralise le problème NP-difficile de couverture par les tests. Il se modélise comme un problème d'identification de couleur d'arêtes dans les hypergraphes. Nous avons étudié la complexité du problème et montré des bornes sur la solution optimale en ajoutant diverses contraintes sur la coloration des arêtes ou sur la structure de l'hypergraphe.

Les bornes sur les solutions optimales des problèmes étudiées sont larges et il est nécessaire d'approfondir l'étude structurelle. Les *designs* semblent être de bons candidats, avec une structure régulière des arêtes. Des résultats structurels permettraient la mise en place de nouvelles heuristiques pour la sélection d'attributs binaires, mais aussi des algorithmes pour générer de nouveaux attributs à partir d'attributs déjà présents dans la base.

Tout au long de cette thèse nous avons traité des problèmes d'exploration de données avec le point de vue de la recherche opérationnelle : analyse de complexité, bornes sur la solution optimale, mise en place d'heuristiques ou de méthodes de résolution exactes. Ces travaux s'inscrivent dans la lignée des travaux de Hammer *et al.* [12, 42], Hansen *et al.* [3, 43], Olafsson *et al.* [61, 73], Mangasarian *et al.* [15, 65, 66]. Les liens entre exploration de données et recherche opérationnelle restent néanmoins à approfondir avec, par exemple, l'utilisation de techniques d'apprentissage dans des heuristiques d'optimisation.

Les résultats de la théorie de Vapnik ainsi que leurs extensions récentes à l'analyse combinatoire de données permettent d'établir des bornes su-

périeures sur l'erreur en généralisation des modèles de l'ACD à partir de l'erreur sur les données d'entraînement. Cette borne peut être utilisée comme objectif d'un programme mathématique permettant de générer des modèles de l'ACD. La mise en place d'une heuristique ou d'une méthode de résolution exacte permettrait d'obtenir de nouveaux modèles.

Annexe A

Définitions

Ce chapitre contient de brefs rappels de théorie des graphes et de complexité. Les définitions et notations utilisées sont issues d'ouvrages classiques [7, 31, 35, 84, 89].

A.1 Graphes

Un **graphe** est un couple $G = (V, E)$ d'ensembles, où E est un ensemble de paires non ordonnées d'éléments de V. Les éléments de V sont les **sommets** du graphe et les éléments de E sont les **arêtes**. Les sommets constituants une arête sont ses **extrémités**. On appelle **boucle**, une arête ayant les deux mêmes extrémités. Un graphe sans boucle est un graphe **simple**.

Deux sommets u et v sont **adjacents** (ou **voisins**) si l'arête (u, v) appartient à E. Le **voisinage** d'un sommet u est le sous-ensemble des sommets qui lui sont adjacents. Le **degré** d'un sommet est la cardinalité de son voisinage.

Le graphe $H = (V', E')$ est un **sous-graphe** du graphe $G = (V, E)$ si $V' \subseteq V$ et $E' \subseteq E$. Le sous-graphe H est **induit** si il contient toutes les arêtes (u, v) de E telles que u et v soient dans V'.

Le graphe **complémentaire** $\overline{G}$, d'un graphe $G = (V, E)$, est le graphe simple défini par $\overline{G} = (V, E')$ où E' est l'ensemble des paires de sommets distincts qui ne sont pas adjacents dans G.

Une **chaîne** est un graphe $P = (V, E)$ avec pour ensemble de sommets $V = \{x_0, x_1, ..., x_n\}$ et pour arêtes $E = \{(x_0, x_1), (x_1, x_2), ...(x_{n-1}, x_n)\}$. Un **cycle** est une chaîne à laquelle on ajoute l'arête (x_n, x_0).

Un **arbre** est un graphe qui n'admet pas de cycle comme sous-graphe induit. Une **étoile** est un arbre avec au plus un sommet de degré supérieur à 1. Un graphe est **biparti** si ses sommets peuvent se partitionner en deux ensembles A et B tels que toutes les arêtes du graphe aient une extrémité dans A et l'autre dans B. Une **clique** ou **graphe complet** est un graphe simple où tous les sommets sont deux à deux adjacents.

Un **hypergraphe** est un couple $H = (V, E)$ où E est un sous-ensemble

des parties de V. Un graphe est donc un hypergraphe particulier où toutes les arêtes sont de taille deux.

A.2 Algorithmes

Un **problème** est une question à laquelle on souhaite apporter une réponse, possédant des paramètres dont les valeurs ne sont pas fixées. Un problème est donc défini par une description de ses paramètres et les propriétés que doivent satisfaire les **solutions**. Une **instance** d'un problème est obtenue en spécifiant les valeurs des paramètres du problème. Par exemple, le problème de tri est décrit par une liste de nombres et une solution est une séquence ordonnée croissante de ces nombres.

Les **problèmes de décision** sont des problèmes auxquels il ne peut exister que deux solutions "vrai" ou "faux". Par exemple, le problème qui demande si un graphe G est un arbre est un problème de décision. Les **problèmes d'optimisation** consistent à trouver la valeur optimale (minimum ou maximum) d'une fonction objectif. Par exemple, le problème qui demande de trouver dans un graphe G le sommet de plus fort degré est un problème d'optimisation.

Un **algorithme** est une procédure de résolution pas à pas d'un problème. Un algorithme possède des entrées, par exemple l'instance d'un problème, et des sorties, par exemple la solution de l'instance. Le temps d'exécution d'un algorithme est le nombre maximal d'opérations élémentaires nécessaires à la terminaison de l'algorithme.

Un algorithme est **polynomial** si son temps d'exécution est polynomial en la taille de son entrée. Par extension, un problème est **polynomial** si il existe un algorithme polynomial permettant de le résoudre. On dit alors que le problème appartient à la classe **P**.

La classe **NP** (*nondeterministic polynomial*) est la classe des problèmes de décision pour lesquels on peut vérifier en temps polynomial que la solution d'une instance est "vrai". Par exemple, le problème de décision qui demande si un graphe G admet comme sous-graphe induit une clique de taille k appartient à NP. En effet, pour montrer qu'une instance admet comme solution "vrai", il suffit d'exhiber un sous-graphe induit qui est une clique avec au moins k sommets. Un problème de P est trivialement dans NP, par contre la réciproque est une question ouverte fondamentale.

On dit qu'un problème de décision Π se **réduit** à un problème de décision Π' s'il existe une fonction g qui transforme une instance x de Π en une instance $g(x)$ de Π', telle que :

- la fonction g est calculable en temps polynomial,
- il existe une réponse "vrai" à l'instance x de Π si et seulement si il existe une réponse "vrai" à l'instance $g(x)$ de Π'.

Un problème de décision Π est **NP-complet** s'il appartient à NP et si

tout problème de NP peut se réduire à Π. Un problème de minimisation Π muni de la fonction objectif f à valeurs dans $\mathbb{Z}$ peut être associé à un problème de décision Π_k dont la solution est "vrai" si et seulement si $f \leq k$. Dans le cas d'un problème de maximisation, l'inégalité est inversée. Un problème d'optimisation est **NP-difficile** si le problème de décision associé est NP-complet.

Un problème de minimisation Π est r-**approximable** si il existe un algorithme polynomial retournant, pour toute instance de Π, une solution x telle que $f(x) \leq rf(x^*)$, avec x^* une solution optimale. Dans le cas d'un problème de maximisation, l'inégalité est inversée.

Annexe B

Sous-graphe dense à k sommets : une bibliographie commentée

Le problème du sous-graphe le plus dense demande de trouver pour un graphe $G = (V, E)$, le sous-ensemble $X \subseteq V$ qui maximise la densité du sous-graphe induit $G[X]$ (voir définition du Chapitre 4). Ce problème est polynomial et se ramène à la résolution de problèmes de flot [36] ou d'un programme linéaire [18]. Lorsque le nombre de sommets k du sous-graphe est fixé le problème est équivalent à trouver le sous-graphe induit avec le plus grand nombre d'arêtes. Si k fait partie de l'entrée du problème alors il s'agit d'une généralisation du problème NP-difficile de la clique maximum [22].

La complexité du problème consistant à trouver le sous-graphe le plus dense à k sommets a été étudiée pour les classes de graphes admettant un algorithme polynomial permettant de trouver la clique maximum. Le problème est trivial pour les arbres (chaque sous graphe connexe à k sommets possède $k-1$ arêtes). Il admet un algorithme polynomial pour les cographes, les *split graphs* [22]. En revanche, le problème reste difficile pour les graphes bipartis, les graphes de comparabilité et les graphes cordaux [22].

Les graphes d'intervalles sont définis comme les graphes d'intersections d'intervalles de la droite réelle. Ils sont une sous-classe des graphes cordaux [31]. La complexité du problème de sous-graphe le plus dense à k sommets est encore ouverte sur cette classe et sur les graphes d'intervalles unitaires (intersection d'intervalles unitaires [37]).

B.1 Un problème approximable dans les graphes d'intervalles

Arora *et al.* [6] proposent un PTAS pour les instances denses du problème ($k = O(n)$ et $m = O(n^2)$) reposant sur la programmation semi-définie. Liazi *et al.* [62, 63] donnent un algorithme polynomial pour les graphes cordaux ayant une chaîne comme graphe d'intersection de ses cliques maximales ainsi qu'une 3-approximation pour les graphes cordaux.

Récemment, Backer et Keil [8] ont proposé une $\frac{3}{2}$-approximation pour les graphes d'intervalles unitaires. Cet algorithme d'approximation repose sur la représentation géométrique des intervalles et utilise une technique de partition de l'instance en segments de longueur fixe. Les auteurs terminent leur papier sur une possible extension de leurs résultats. Leur algorithme peut être étendu à une $\frac{\tau}{\tau-1}$-approximation à condition de savoir résoudre le problème sur les graphes d'intervalles unitaires représentables sur un segment de longueur $\tau \in \mathbb{N}$. En analysant la preuve utilisée, nous avons remarqué que l'utilisation d'un PTAS au lieu d'un algorithme exact amènerait à une $(1-\epsilon)\frac{\tau}{\tau-1}$-approximation. Pour les graphes d'intervalles unitaires représentables sur un segment de longueur k, il est possible de montrer que soit il existe une clique de taille supérieure à k et donc une solution triviale, soit que le graphe vérifie les propriétés requises par le PTAS de Arora *et al* [6]. On en déduit l'existence d'un PTAS pour les graphes d'intervalles unitaires. Une approche similaire a été utilisée indépendamment par Chen *et al.* [19] pour généraliser le résultat aux graphes d'intersections de disques unitaires.

Les dernières avancées sur le sujet sont les résultats de Nonner [72]. Il donne un PTAS reposant sur un programme dynamique pour le problème du sous-graphe le plus dense à k sommets dans les graphes d'intervalles. Contrairement à l'algorithme d'approximation pour les graphes d'intervalles unitaires, cet algorithme est purement combinatoire.

B.2 Un problème NP-difficile ?

La conjecture actuelle est que ce problème est NP-difficile [72]. Il n'existe que quelques problèmes difficiles pour les graphes d'intervalles. En particulier, l'arrangement linéaire optimal semble assez proche pour tenter une réduction. Dans ce problème, on cherche un étiquetage L des sommets avec les entiers de 1 à n qui minimise la fonction :

$$W(G, L) = \sum_{\{u,v\} \in E} |L(u) - L(v)|$$

Intuitivement les k étiquettes de l'étiquetage optimal du sous-graphe le plus dense à k sommets doivent être proches. L'utilisation d'un nombre

polynomial d'appels à une procédure permettant de calculer le sous-graphe le plus dense à k sommets pourrait permettre la reconstruction de l'étiquetage optimal.

La complexité du problème du sous-graphe le plus dense à k sommets dans les graphes d'intervalles reste ouverte. Les résultats récents ont permis de montrer l'approximabilité du problème dans cette classe de graphes. Les versions pondérées du problème sont difficiles dans les graphes d'intervalles, même unitaires. Ces problèmes sont connus dans la littérature sous le nom du problème de sac à dos quadratique.

Bibliographie

[1] G. Alexe, S. Alexe, P. L. Hammer et A. Kogan : Comprehensive vs. comprehensible classifiers in logical analysis of data. *Discrete Applied Mathematics*, 156:870–882, 2008.

[2] G. Alexe et P. L. Hammer : Spanned patterns for the logical analysis of data. *Discrete Applied Mathematics*, 154:1039–1049, 2006.

[3] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron et L. Liberti : Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4):046112, 2010.

[4] D. Aloise, A. Deshpande, P. Hansen et P. Popat : NP-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[5] M. Anthony : Generalization error bounds for Logical Analysis of Data. Rapport technique, Rutcor Research Report, 2011.

[6] S. Arora, D. Karger et M. Karpinski : Polynomial time approximation schemes for dense instances of NP-hard problems. *Dans Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 284–293. ACM, 1995.

[7] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi et V. Kann : *Complexity and Approximation : Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[8] J. Backer et J. M. Keil : Constant factor approximation algorithms for the densest k-subgraph problem on proper interval graphs and bipartite permutation graphs. *Information Processing Letters*, 110(16):635–638, 2010.

[9] R. E. Bixby : Solving Real-World Linear Programs : A Decade and More of Progress. *Operations Research*, 50:3–15, 2002.

[10] T. O. Bonates, P. L. Hammer et A. Kogan : Maximum patterns in datasets. *Discrete Applied Mathematics*, 156:846–861, 2008.

[11] E. Boros, P. L. Hammer, T. Ibaraki et A. Kogan : Logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 2000.

[12] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz et I. Muchnik : An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):292–306, 2000.

[13] E. Boros, T. Horiyama, T. Ibaraki, K. Makino et M. Yagiura : Finding essential attributes in binary data. *Dans Intelligent Data Engineering and Automated Learning — IDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents*, volume 1983 de *Lecture Notes in Computer Science*, pages 211–224. Springer Berlin / Heidelberg, 2000.

[14] B. E. Boser, I. M. Guyon et V. N. Vapnik : A training algorithm for optimal margin classifiers. *Dans* D. Haussler, éditeur : *Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT '92), July 27-29, 1992, Pittsburgh, PA, USA*, pages 144–152. ACM Press, New York, NY, USA, 1992.

[15] P. S. Bradley, O. L. Mangasarian et W. N. Street : Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998.

[16] U. Brandes, D. Delling, M. Höfer, M. Gaertler, R. Görke, Z. Nikoloski et D. Wagner : On finding graph clusterings with maximum modularity. *Dans Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'07)*, Lecture Notes in Computer Science. Springer, 2007.

[17] L. Breiman, J. Friedman, C. J. Stone et R. A. Olshen : *Classification and Regression Trees*. Chapman and Hall/CRC, 1 édition, 1984.

[18] M. Charikar : Greedy approximation algorithms for finding dense components in a graph. *Dans APPROX*, pages 84–95, 2000.

[19] D. Chen, R. Fleischer et J. Li : Densest k-subgraph approximation on intersection graphs. *Approximation and Online Algorithms*, 6534:83–93, 2011.

[20] F. R. K. Chung : *Spectral Graph Theory*. American Mathematical Society, 1997.

[21] A. Clauset, C. Moore et M. E. J. Newman : Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008.

[22] D. G. Corneil et Y. Perl : Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9:27–39, 1984.

[23] A. Cornuéjols et L. Miclet : *Apprentissage artificiel : Concepts et algorithmes*. Eyrolles, 2003.

[24] Y. Crama, P. L. Hammer et T. Ibaraki : Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research*, 16:299–325, 1988.

[25] Y. Crama et P.L. Hammer : *Boolean Functions : Theory, Algorithms and Applications*. Cambridge University Press, 2011.

[26] W. H. Cunningham : Optimal attack and reinforcement of a network. *Journal of the ACM*, 32(3):549–561, 1985.

[27] J. Darlay : Analyse combinatoire de données et analyse de temps de survie. Rapport de stage master 2 recherche, Grenoble Institute of Technology, 2008.

[28] S. Dasgupta : The hardness of k-means clustering. Rapport technique, University of California, 2008.

[29] M. Dash, H. Liu et H. Motoda : Consistency based feature selection. *Dans Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, PADKK '00, pages 98–109, London, UK, 2000. Springer-Verlag.

[30] I. Dhillon, Y. Guan et B. Kulis : Weighted graph cuts without eigenvectors : a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[31] R Diestel : *Graph Theory*, volume 173 de *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, 2005.

[32] T. G. Dietterich et G. Bakiri : Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[33] R. A. Fisher : The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936.

[34] T. R. Fleming et D. P. Harrington : *Counting processes and survival analysis*. John Wiley & Sons, Ltd., New York, 1991.

[35] M. R. Garey et D. S. Johnson : *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman & Co Ltd, 1979.

[36] A. V. Goldberg : Finding a maximum density subgraph. Rapport technique UCB/CSD-84-171, EECS Department, University of California, Berkeley, 1984.

[37] M.C. Golumbic : *Algorithmic graph theory and perfect graphs*. Academic Press, 1980.

[38] E. Graf, C. Schmoor, W. Sauerbrei et M. Schumacher : Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medecine*, 18:2529–2545, 1999.

[39] S. Grivaud Martin, L.-P. Kronek, D. Valeyre, N. Brauner, P.-Y. Brillet, H. Nunes, M. W. Brauner et F. Réty : High-resolution computed tomography to differentiate chronic diffuse interstitial lung diseases with predominant ground-glass pattern using logical analysis of data. *European Radiology*, 20:1297–1310, 2010.

[40] I. Guyon et A. Elisseeff : An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[41] M. A. Hall : *Correlation-based Feature Subset Selection for Machine Learning*. Thèse de doctorat, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1999.

[42] P. Hammer et T. Bonates : Logical analysis of data—an overview : From combinatorial optimization to medical applications. *Annals of Operations Research*, 148(1):203–225, 2006.

[43] P. Hansen et C. Meyer : A new column generation algorithm for logical analysis of data. *Annals of Operations Research*, pages 1–35, 2011.

[44] F. E. Harrel, K. L. Lee, R. M. Califf, D. B. Pryor et R. A. Rosati : Regression modelling strategies for improved prognostics. *Statistics in Medecine*, 3(2):143–152, 1984.

[45] J. A. Hartigan et M. A. Wong : A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[46] H. Ishwaran et U. B. Kogalur : Random survival forests for R. *R News*, 7/2:25–31, 2007.

[47] P. Jaccard : Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[48] E. L. Kaplan et P. Meier : Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282): 457–481, 1958.

[49] M. G. Karpovsky, K. Chakrabarty et L. B. Levitin : On a new class of codes for identifying vertices in graphs. *IEEE Transactions on Information Theory*, 44:599–611, 1998.

[50] B. W. Kernighan et S. Lin : An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(1):291–307, 1970.

[51] J. P. Klein et M. L. Moeschberger : *Survival Analysis : Techniques for Censored and Truncated Data*. Springer, second édition, 2003.

[52] J. Kleinberg : An impossibility theorem for clustering. *Dans* S. Becker, S. Thrun et K. Obermayer, éditeurs : *Advances in Neural Information Processing Systems*, pages 446–453. MIT Press, 2002.

[53] D. König : Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. (german). *Mathematische Annalen*, 77:453–465, 1916.

[54] A. Kogan : On the essential test sets of discrete matrices. *Discrete Applied Mathematics*, 60:249–255, 1995.

[55] D.A. Koonce et S.-C. Tsai : Using data mining to find patterns in genetic algorithm solutions to a job shop schedule. *Computers & Industrial Engineering*, 38:361–374, 2000.

[56] L.-P. Kronek : *Analyse combinatoire de données appliquée à la recherche médicale*. Thèse de doctorat, Grenoble Institute of Technology, 2008.

[57] L.-P. Kronek et A. Reddy : Logical analysis of survival data : prognostic survival models by detecting high-degree interactions in right-censored data. *Bioinformatics*, 24(16):248–253, 2008.

[58] M. LeBlanc et J. Crowley : Relative risk tree for censored survival data. *Biometrics*, 48(2):411–425, 1992.

[59] M. Lejeune et F. Margot : Optimization for Simulation : LAD Accelerator. To appear in Annals of Operations Research.

[60] P. Lemaire, N. Brauner, P. Hammer, C. Trivin, J.-C. Souberbielle et R. Brauner : Improved screening for growth hormone deficiency using logical analysis of data. *Medical Science Monitor*, 15:MT5–10, 2009.

[61] X. Li et S. Olafsson : Discovering dispatching rules using data mining. *Journal of Scheduling*, 8:515–525, 2005.

[62] M. Liazi, I. Milis, F. Pascual et V. Zissimopoulos : The densest k-subgraph problem on clique graphs. *Journal of combinatorial optimization*, 14(4):465–474, 2007.

[63] M. Liazi, I. Milis et V. Zissimopoulos : A constant approximation algorithm for the densest k-subgraph problem on chordal graphs. *Information Processing Letters*, 108(1):29–32, 2008.

[64] J. B. Macqueen : Some Methods for classification and analysis of multivariate observations. *Dans Procedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[65] O. L. Mangasarian : Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

[66] O. L. Mangasarian : Support vector machine classification via parameterless robust linear programming. *Optimization Methods and Software*, 20(1):115–125, 2005.

[67] G. E. Moore : Cramming more components onto integrated circuits. *Electronics*, 38:114–117, 1965.

[68] M. E. J. Newman : The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[69] M. E. J. Newman : Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004.

[70] M. E. J. Newman : Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[71] M. E. J. Newman et M. Girvan : Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[72] T. Nonner : PTAS for densest k-subgraph in interval graphs. Rapport technique, IBM Research - Zurich, 2011.

[73] S. Olafsson, X. Li et S. Wu : Operations research and data mining. *European Journal of Operational Research*, 187:1429–1448, 2008.

[74] R. W. Payne et D. A. Preece : Identification keys and diagnostic tables : A review. *Journal of the Royal Statistical Society. Series A (General)*, 143(3):253–292, 1980.

[75] L. Pitt et L. G. Valiant : Computational limitations on learning from examples. *Journal of the ACM*, 35:965–984, 1988.

[76] M. Preissmann et A. Sebö : Graphic submodular function minimization : A graphic approach and applications. *Dans Research Trends in Combinatorial Optimization*, pages 365–385. Springer Berlin Heidelberg, 2009.

[77] J. R. Quinlan : Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[78] J. R. Quinlan : *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, 1 édition, 1992.

[79] R Development Core Team : *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. ISBN 3-900051-07-0.

[80] B. D. Ripley et R. M. Ripley : Neural networks as statistical methods in survival analysis. *Artificial Neural Networks : Prospects for Medecine*, 1998.

[81] H. S. Ryoo et I.-Y. Jang : MILP approach to pattern generation in logical analysis of data. *Discrete Applied Mathematics*, 157:749–761, 2009.

[82] W. Sauerbrei et P. Royston : Building multivariable prognostic and diagnostic models : transformation of the predictors by using fractional polynomials. *Journal of The Royal Statistical Society Series A*, 127(1): 71–94, 1999.

[83] S. Schaeffer : Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[84] A. Schrijver : *Combinatorial optimization : polyhedra and efficiency*. Springer, 2003.

[85] J. Shi et J. Malik : Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[86] J. Síma et S. E. Schaeffer : On the NP-completeness of some graph cluster measures. *Dans SOFSEM 2006 : Theory and Practice of Computer Science*, volume 3831 de *Lecture notes in computer science*, pages 530–537, 2006.

[87] M. Stoer et F. Wagner : A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997.

[88] S. Tufféry : *Data Mining et Statistique décisionnelle l'intelligence des données*. Éditions Technip, 2005.

[89] J. H. van Lint et R. M. Wilson : *A Course in Combinatorics*. Cambridge University Press, 1992.

[90] V. N. Vapnik : *Statistical Learning Theory*. Wiley-Interscience, 1998.

[91] S. Wang et J. M. Siskind : Image segmentation with ratio cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):675–690, 2003.

[92] I. H. Witten et E. Frank : *Data Mining : Practical Machine Learning Tools and Techniques with Java Implementations (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 1st édition, 1999.

[93] D.H. Wolpert : On the connection between in-sample testing and generalization error. *Complex Systems*, 6:47–94, 1992.

[94] W. W. Zachary : An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[95] B. Zupan, J. Demsar, M. W. Kattan, J. R. Beck et I. Bratko : Machine learning for survival analysis : a case study on recurrence of prostate cancer. *Artificial intelligence in medicine*, 20(1):59–75, 2000.

Index