UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Fonctions latticielles polynomiales pour l'interpolation et la classification monotone

THÈSE

présentée et soutenue publiquement le 29 janvier 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Quentin Brabant

Composition du jury

Président :	Jean-Luc Marichal	Université du Luxembourg
Rapporteurs :	Sébastien Destercke	Université de Technologie de Compiègne, CNRS
	Alexis Tsoukiàs	Université Paris Dauphine, LAMSADE, CNRS
Examinateurs :	Hélène Fargier	Université Paul Sabatier, IRIT, CNRS
	Claire Gardent	CNRS, LORIA
Invités :	Didier Dubois	Université Paul Sabatier, IRIT, CNRS
	Michel Grabisch	Université Paris 1 Panthéon-Sorbonne
	Henri Prade	Université Paul Sabatier, IRIT, CNRS
Directeur :	Miguel Couceiro	Université de Lorraine, CNRS, Inria, LORIA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Un grand merci, tout d'abord, à mon directeur, Miguel Couceiro, qui s'est impliqué avec énergie dans l'encadrement de ma thèse, durant 3 ans (et demi), et a tenté de me transmettre la hargne du mathématicien. Merci ensuite au jury qui a lu, écouté et commenté mon travail. Merci en particulier à mes deux rapporteurs pour le temps investi dans la lecture de mon manuscrit.

Je tiens également à remercier un certain nombre de personnes, avec lesquelles j'ai pu travailler courtement et/ou qui ont apporté de l'eau à mon moulin sous forme de remarques, idées et contradictions. Merci donc à : Henri Prade, Didier Dubois, Agnès Rico, Tamas Waldhauser, Bruno Zanutini, Fabien Labernia, José Rui Figueira, Amedeo Napoli, Justine Reynaud, Aurore Alcolei, Chedy Raïssi. Merci à Gabin Personeni d'avoir partagé son expertise des opérateurs de treillis et son implémentation des pattern structures.

Mes remerciements ne seraient pas complets s'ils n'intégraient pas tous les gens sur qui j'ai pu m'appuyer au quotidien, et sans qui j'aurais sûrement baissé les bras avant la fin. Merci à mes colocataires passés et présents, pour tous les tours de ménages qu'ils ont fait à ma place. Merci à Jean-Ludovique, à qui je dois toujours 30 euros et qui, j'en suis sûr, ne m'en veut pas. Merci à mes amies russophones, qui sont toujours vachement sympa. Merci à mes amis francophones, aussi. Merci à mes parents qui me laissent revenir à la maison quand je disparais sans donner de nouvelles pendant 2 mois.

Merci à ma moitié dont je mérite à peine le quart.

Table des matières

Introduction

Cette thèse porte sur les Fonctions Latticielles Polynomiales (FLP), c'est à dire les fonctions pouvant être exprimées à l'aide des opérateurs de treillis, et sur les intégrales de Sugeno, qui appartiennent à la classe des FLP. Dans une première partie nous étudions les FLP d'un point de vue théorique. Ces études débouchent sur des algorithmes permettant de calculer une FLP compatible avec un ensemble d'exemples de la forme (arguments, résultat) en temps polynomial. Dans une deuxième partie, nous étudions les applications des intégrales de Sugeno à la tâche de classification monotone.

1 L'Aide à la Décision Multi Critères

Ces travaux sont à l'origine motivés par l'approche qualitative de la modélisation des préférences en *Aide à la Décision Multi Critères* [56] (*Multiple Criteria Decision Aiding/Analysis, MCDA*). La MCDA est une sous-discipline de la Recherche Opérationnelle, dans laquelle on considère des problèmes de décision impliquant un ensemble d'alternatives ainsi qu'un ensemble de critères selon lesquels chaque alternative est évaluée. Une des problématiques centrales de la MCDA est celle de la modélisation d'une relation représentant les préférences du décideur sur l'ensemble des alternatives, en fonction des évaluations de chaque alternative selon chaque critère.

Représentons chaque alternative par un tuple de la forme $\mathbf{x} = (x_1, \ldots, x_n)$, où x_i est l'évaluation de l'alternative $\mathbf{x}$ selon le i-ème critère. On appelle n le nombre de critères considérés. Une *relation de préférence* est un ordre (ou quasi-ordre) dénoté par $\preccurlyeq$ et tel que, pour deux alternatives $\mathbf{x}$ et $\mathbf{x'}$, l'inégalité $\mathbf{x} \preccurlyeq \mathbf{x'}$ signifie « $\mathbf{x'}$ est au moins aussi désirable que $\mathbf{x}$ ». Afin de modéliser une relation de préférence, une approche courante est de faire appel à une *fonction d'utilité* U, qui associe à chaque alternative une évaluation globale, en fonction de ses évaluations sur chaque critère. Ainsi, on détermine la relation de préférence $\preccurlyeq$ d'après une fonction d'utilité U, par

$$\mathbf{x} \preccurlyeq \mathbf{x'} \quad \Longleftrightarrow \quad U(\mathbf{x}) \leq U(\mathbf{x'})$$

pour toutes alternatives $\mathbf{x}$ et $\mathbf{x'}$.

Remarquez que les évaluations peuvent être effectuées sur des échelles ordinales non numériques, et/ou sur plusieurs échelles différentes, et que la nature des échelles utilisées conditionne en partie le choix de la fonction d'utilité. Un cas particulier considéré dans la littérature est celui où toutes les évaluations sont effectuées sur une échelle commune L, qui est un ensemble totalement ordonné et borné. Dans ce cas, les fonctions d'utilité typiquement considérées sont appelées *fonctions d'agrégations* [74].

2 Fonctions d'agrégation

On dénote l'élément minimal et l'élément maximal de L par 0 et 1, respectivement. Une *fonction d'agrégation* [55] est une fonction $f : L^n \to L$ qui

- est *non-décroissante* : pour tout $x_1, \ldots, x_n, x'_1, \ldots, x'_n \in L$

$$\forall i \in \{1, \ldots, n\}, x_i \leq x'_i \quad \Longrightarrow \quad f(x_1, \ldots, x_n) \leq f(x'_1, \ldots, x'_n),$$

- vérifie les conditions limites $f(0, \ldots, 0) = 0$ et $f(1, \ldots, 1) = 1$.

La fonction d'agrégation la plus basique et la mieux connue est certainement la moyenne arithmétique pondérée, donnée par

$$M(x_1, \ldots, x_n) = \frac{1}{n} \sum_{i=1}^{n} w_i * x_i,$$

où $w_1, \ldots, w_n$ sont les poids affectés aux paramètres (et dont la somme vaut 1). Toutefois, une fonction linéaire des évaluations $x_1, \ldots, x_n$ ne permet pas toujours de rendre compte fidèlement des préférences de l'utilisateur [53].

Les intégrales des Choquet sont des fonctions d'agrégation plus expressives, qui ont suscité l'intérêt de la communauté MCDA dans les dernières décennies [54]. Elles généralisent la notion de moyenne arithmétique pondérée en accordant un poids à chaque sous-ensemble de critères. Une intégrale de Choquet est une fonction $C : [0, 1]^n \to [0, 1]$ de la forme

$$C(x_1, \ldots, x_n) = \sum_{I \subseteq \{1, \ldots, n\}} w_I * \bigwedge_{i \in I} x_i,$$

où w_I est un poids propre au sous-ensemble de critères $I \subseteq [n]$, qui est donné par la *transformée de Möbius* d'une *capacité* (ces notions seront définies dans la Section 5.4). Les intégrales de Choquet ont pour avantage d'être plus expressives que les moyennes arithmétiques pondérées, tout en étant compréhensibles par un décideur et non-décroissantes sur chaque paramètre.

Considérons maintenant le cas où L est une échelle ordinale non numérique (aussi dite *qualitative*), par exemple l'ensemble {mauvais, moyen, bon, très bon} ordonné de la manière suivante :

$$\text{mauvais} < \text{moyen} < \text{bon} < \text{très bon}.$$

Sur une telle échelle, l'addition et la multiplication ne sont pas définies. Afin de modéliser les préférences du décideur par une fonction d'utilité, deux principales approches sont possibles. La première consiste à convertir les valeurs de l'échelle ordinale en des valeurs numériques, pour faire ensuite appel à une fonction d'utilité basée sur les opérations arithmétiques comme l'intégrale de Choquet (voir par exemple la méthode TOMASO [77, 87]). Une seconde approche, dite *qualitative*, est d'avoir recours à une fonction d'agrégation sur L dont la définition est basée sur des notions purement ordinales. Les intégrales de Sugeno sont de telles fonctions.

3 Modélisation qualitative des préférences avec les FLP

Soit une ensemble totalement ordonné L. Une FLP sur L est une fonction $p : L^n \to L$ pouvant être exprimée à partir de variables, de constantes, et des opérateurs binaires $\wedge$ (minimum) et

$\vee$ (maximum). Par exemple, dans le cas où $L = \{a, b, c, d\}$ et $a < b < c < d$, les fonctions $p, q, r : L^n \to L$ définies par

$$\begin{aligned} p(x_1) &= b \vee x_1, \\ q(x_1, x_2, x_3) &= x_1 \wedge (x_2 \vee x_3), \\ r(x_1, x_2, x_3) &= (c \wedge x_1) \vee (b \wedge x_2 \wedge x_3), \end{aligned}$$

sont des FLP sur L. Remarquez que, les opérations $\wedge$ et $\vee$ étant non-décroissantes, toute FLP est également non décroissante. L'intersection des FLP et des fonctions d'agrégation sur L est une classe de fonctions appelées *intégrales de Sugeno*. Toute intégrale de Sugeno peut être exprimée sous la forme

$$S(x_1, \ldots, x_n) = \bigvee_{I \subseteq \{1,\ldots,n\}} \Big(w_I \wedge \bigwedge_{i \in I} x_i \Big),$$

où les valeurs de w_I pour chaque $I \subseteq [n]$ sont déterminées par une capacité. Remarquez que, d'une manière similaire aux intégrales de Choquet, le résultat d'une intégrale de Sugeno dépend des valeurs w_I associées à chaque sous-ensemble de critères $I \subseteq [n]$. Les intégrales de Sugeno sont parfois vues comme des analogues ordinales des intégrales de Choquet et constituent la plus large classe de fonctions d'agrégation pouvant être exprimées à l'aide des opérateurs $\wedge$ et $\vee$. Pour ces raisons, il a été proposé (voir par exemple [41, 42, 52]) d'utiliser l'intégrale de Sugeno en tant que fonction d'utilité pour la modélisation qualitative de préférences.

Notez cependant que si L est une échelle finie et de petite cardinalité (comme cela est typiquement le cas des échelles qualitatives), une fonction d'utilité valuée sur L produira nécessairement un grand nombre de paires d'alternatives « ex-aequo ». On peut en fait considérer que, dans un tel cas, l'approche qualitative poursuit un objectif différent de celui de l'approche numérique de la modélisation des préférences. Tandis que les fonctions d'utilité valuées sur des intervalles réels permettent une description fine des préférences, où la plupart des paires d'alternatives peuvent être départagées, les fonctions d'utilité valuées sur des ensembles finis peuvent être vues comme des fonctions de classification.

4 Les FLP définies sur les treillis

Un *treillis* est un ensemble ordonné L dans lequel toute paire d'éléments $a, b \in L$ possède un plus grand minorant commun, appelé infimum de a et b et un plus petit majorant commun, appelé supremum de a et b. Les ensembles totalement ordonnés sont des cas particuliers de treillis, dans lesquels l'infimum correspond au minimum et le supremum au maximum. On dénote les opérateurs d'infimum et de supremum par $\wedge$ et $\vee$, respectivement. Une *fonction latticielle polynomiale (FLP)* sur un treillis L est une fonction qui peut être exprimée à partir de variables, de constantes et des opérateurs $\wedge$ et $\vee$.

Dans le cadre de la modélisation des préférences, les treillis peuvent représenter des échelles d'évaluation où certaines valeurs sont incomparables. De telles échelles peuvent notamment intégrer des valeurs représentant l'indifférence ou l'incertitude (un exemple fictif illustrant cette possibilité sera développé dans la Sous-section 3.6.3).

Dans la première partie de la cette thèse, nous étudions les FLP définies sur des treillis.

Nous traitons principalement le problème d'interpolation, qui est celui de déterminer s'il existe une FLP correspondant exactement à un ensemble d'exemples de la forme (arguments, résultat) représentés par une fonction partielle $f : D \to L$, où $D \subseteq L^n$ et, le cas échéant, de calculer une telle FLP.

5 La classification monotone

La *classification monotone* est un cas particulier de la tâche de classification, où les valeurs d'attributs ainsi que les classes appartiennent à des ensembles totalement ordonnés, et où l'on cherche à apprendre un classifieur monotone. Cette tâche peut être effectuée à l'aide de méthodes de classification classiques, auxquelles on ajoute certaines contraintes qui garantissent la monotonie du classifieur, ou bien à l'aide de modèles et algorithmes pensés spécifiquement pour elle.

Une des manières d'aborder cette tâche est de s'appuyer sur les fonctions d'utilité utilisées en MCDA. Une telle approche a déjà été considérée dans le cas de l'intégrale de Choquet (voir [93]).

La seconde partie de cette thèse sera consacrée à l'application des intégrales des Sugeno à la tâche de classification monotone. Les Fonctions d'Utilité de Sugeno (FUS) [25] sont des généralisations des intégrales de Sugeno qui permettent de fusionner des valeurs venant d'échelles différentes, tout en conservant la nature purement ordinale du modèle d'origine. Une des difficultés faisant obstacle à l'application de ces fonctions est leur manque d'expressivité ; en effet, certaines fonctions monotones ne correspondent à aucune FUS (pour une étude plus complète de l'expressivité relative des FUS et des règles de décision, voir [58, 12]), et certaines propriétés des FUS, qui seront exposées dans le Chapitre 5, suggèrent que ces fonctions n'ont pas l'expressivité nécessaire pour modéliser fidèlement des données empiriques. Afin de surmonter ce problème, nous étudions un modèle introduit dans [23], qui consiste en un maximum (ou un minimum) de plusieurs FUS, et tentons d'évaluer le nombre de FUS nécessaires pour modéliser des données empiriques via ce nouveau modèle.

6 Plan de la thèse

Cette thèse est divisée en deux parties. Pour chaque chapitre dont le contenu a été au moins partiellement publié, nous indiquons les publications correspondante.

Première partie : Nous étudions les FLP définies sur des treillis d'un point de vue théorique, et nous exposons notamment des résultats concernant le problème d'interpolation.

- *Chapitre 1.* Nous présentons les notions de base concernant les ordres partiels et les treillis, qui seront utilisées dans les deux chapitres suivants.
- *Chapitre 2.* Nous considérons les FLP définies sur des treillis finis, et présentons une première méthode pour résoudre le problème d'interpolation en temps polynomial.
- *Chapitre 3.* Nous considérons les FLP définies sur des treillis distributifs bornés. Nous présentons une méthode pour décrire l'ensemble des solutions du problème d'interpolation en temps polynomial, dans le cas où le treillis considéré peut être plongé dans un produit fini de chaînes (publication n° 1 de la liste ci-dessous).
- *Chapitre 4.* Nous présentons une caractérisation de la classe des FLP dites k-maxitives (publications n° 4 et 9).

Deuxième partie : Nous étudions l'utilisation des FUS pour la tâche de classification monotone.

- *Chapitre 5.* Nous présentons la tâche de classification monotone et dressons un état de l'art des méthodes existantes pour cette tâche.

- *Chapitre 6.* Nous étudions les FUS en nous appuyant sur le paradigme des règles de décision. Nous présentons un modèle basé sur le maximum ou le minium de plusieurs FUS, et cherchons à déterminer le nombre minimal de FUS nécessaires à la réalisation de différentes tâches, en particulier la modélisation de données empiriques. Ce chapitre est une extension d'un travail publié (publication n° 2).

Publications

1. Q. Brabant, M. Couceiro, J.R. Figueira, Interpolation by lattice polynomial functions : A polynomial time algorithm,*à paraître dans Fuzzy Sets and Systems.*

2. Q. Brabant, M. Couceiro, D. Dubois, H. Prade, A. Rico. Sugeno Integral for Rule-Based Ordinal Classification. *IJCAI-ECAI-2018 Workshop on Learning and Reasoning : Principles and Applications to Everyday Spatial and Temporal Knowledge*, 23–24, 2018.

3. Q. Brabant, M. Couceiro, D. Dubois, H. Prade, A. Rico. Extracting Decision Rules from Qualitative Data via Sugeno Utility Functionals. *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations (IPMU)*, 253–265. Communications in Computer and Information Science. Springer, Cham, 2018.

4. Q. Brabant, and M. Couceiro. k-maxitive Sugeno Integrals as Aggregation Models for Ordinal Preferences. *Fuzzy Sets and Systems*, 343 : 65—75, 2018.

5. Q. Brabant, M. Couceiro, A. Napoli, J. Reynaud. From Meaningful Orderings in the Web of Data to Multi-Level Pattern Structures. *Foundations of Intelligent Systems (ISMIS)*, 622—631. Lecture Notes in Computer Science. Springer, Cham, 2017.

6. Q. Brabant, M. Couceiro. Apprentissage d'intégrales de Sugeno à partir de données inconsistantes. *25ème Rencontres Francophones Sur La Logique Floue et Ses Applications (LFA)*, 49–56, 2016.

7. Q. Brabant, M. Couceiro, F. Labernia, A. Napoli. A dimensionality reduction approach for qualitative preference aggregation. *International Symposium on Aggregation and Structures*, 2016.

8. Q. Brabant, M. Couceiro, F. Labernia, A. Napoli. Une approche de réduction de dimensionnalité pour l'agrégation de préférences qualitatives. *16ème Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances (EGC)*, 345–350, 2016.

9. Q. Brabant , M. Couceiro. Axiomatisation des intégrales de Sugeno k-maxitives. *24éme Rencontres Francophones Sur La Logique Floue et Ses Applications (LFA)*, 179–186, 2015.

Première partie

Fonctions latticielles polynomiales et interpolation

Chapitre 1

Ordres et treillis

Ce chapitre introduit des notions et notations qui seront utilisées dans les chapitres suivants. Celles-ci concernent principalement les ensembles ordonnés et les treillis. Pour une introduction plus complète à ces notions, se référer, par exemple, à [36].

1.1 Ensembles ordonnés

Relations, ordres et ensembles ordonnés. Soit X un ensemble, et $\leq$ une relation binaire sur X. On dit que $\leq$ est

- *transitive* si, pour tout $x, y, z \in X$,
$$\big[x \leq y \text{ et } y \leq z\big] \quad \Longrightarrow \quad x \leq z,$$
- *réflexive* si, pour tout $x \in X$, on a $x \leq x$,
- et *antisymétrique* si, pour tout $x, y \in X$,
$$\big[x \leq y \text{ et } y \leq x\big] \quad \Longrightarrow \quad x = y.$$

Une relation binaire qui est réflexive, transitive et antisymétrique est appelée *ordre*, ou *ordre partiel*.

Un *ensemble partiellement ordonné*, ou *poset* (abréviation de *partially ordered set*), est un couple $(X, \leq)$ où X est un ensemble appelé *univers* de $(X, \leq)$ et $\leq$ est un ordre sur X. On dit que $(X, \leq)$ est un *ensemble totalement ordonné* ou une *chaîne*, si pour tout $x, y \in X$
$$x \leq y \quad \text{ou} \quad y \leq z.$$

Notations. Lorsque cela ne génère pas d'ambiguïté, on désignera un poset par son univers ; par défaut l'ordre d'un tel poset sera dénoté par $\leq$. On écrira par exemple « X est un poset », pour signifier que X est un ensemble auquel on a associé un ordre dénoté par $\leq$. On utilisera également les notations suivantes : pour tout $x, y \in X$,
$$\begin{aligned}
x \geq y \quad &\text{si} \quad y \leq x,\\
x < y \quad &\text{si} \quad x \leq y \text{ et } x \neq y,\\
x \not\leq y \quad &\text{si} \quad x \leq y \text{ n'est pas vérifiée},\\
x \not< y \quad &\text{si} \quad x < y \text{ n'est pas vérifiée},\\
x \parallel y \quad &\text{si} \quad x \not\leq y \text{ et } y \not\leq x.
\end{aligned}$$

Lorsque $x \parallel y$, on dit que x et y sont *incomparables*. Lorsque X est un ensemble d'ensembles, l'ordre associé par défaut à X sera la relation d'inclusion, dénotée par $\subseteq$. Les relations $\supseteq$, $\subset$, $\not\subseteq$, $\not\subset$ sont définies de manière analogue à $\geq$, $<$, $\not\leq$, $\not<$, respectivement.

Bornes, minimums et maximums. Soit deux ensembles ordonnés X et Y tels que $Y \subseteq X$. Une *borne inférieure* de Y (dans X) est un élément $x \in X$ qui vérifie $x \leq y$, pour tout $y \in Y$. De manière analogue, une *borne supérieure* de Y est un élément $x \in X$ qui vérifie $y \leq x$, pour tout $y \in Y$. On dénote l'ensemble des bornes inférieures de Y dans X par $\inf(Y)$ et l'ensemble des bornes supérieures de Y par $\sup(Y)$.

Les ensembles $\downarrow Y$ et $\uparrow Y$ sont définis par

$$\downarrow Y = \{x \in X \mid \exists y \in Y,\ x \leq y\} \quad \text{et} \quad \uparrow Y = \{x \in X \mid \exists y \in Y,\ y \leq x\}.$$

Lorsque $Y = \{y\}$, on dénotera simplement $\downarrow Y$ par $\downarrow y$ et $\uparrow Y$ par $\uparrow y$.

Pour tout ensemble ordonné X, on définit les ensembles $\min(X)$ et $\max(X)$ de la manière suivante :

$$\min(X) = \{x \in X \mid \forall x' \in X,\ x' \not< x\}, \quad \max(X) = \{x \in X \mid \forall x' \in X,\ x \not< x'\}.$$

Produit Cartésien. Soient X et Y deux ensembles. Le *produit Cartésien* de X et Y est l'ensemble

$$X \times Y = \{(x, y) \mid x \in X, y \in Y\}.$$

Pour tout entier positif n, on définit l'ensemble des tuples de taille n constitués d'éléments de X de la manière suivante :

$$X^n = \begin{cases} X \text{ si } n = 1, \\ X \times X^{n-1} \text{ si } n > 0. \end{cases}$$

Si X et Y sont des ensembles ordonnés, l'ordre associé à $X \times Y$ est le produit des ordres de X et Y donné par :

$$\forall x, x' \in X,\ \forall y, y' \in Y, \quad (x, y) \leq (x', y') \quad \text{si} \quad \big[x \leq x' \text{ et } y \leq y'\big].$$

1.2 Treillis

Un treillis est un ensemble ordonné $(L, \leq)$ tel que, pour tout sous-ensemble fini non vide A de L, $\inf(A)$ possède un élément maximal unique et $\sup(A)$ possède un élément minimal unique. Ces éléments sont appelés respectivement *infimum* et *supremum* de A et dénotés par $\bigwedge A$ et $\bigvee A$.

Exemple 1. La Figure 1.1 donne trois exemples de posets. Les deux premiers ne sont pas des treillis. En effet, dans le premier poset on a, par exemple, $\inf(\{a, b\}) = \emptyset$. Dans le second, on a $\sup(\{a, b\}) = \{c, d\}$; $\{a, b\}$ n'a donc pas de supremum. Le troisième poset est un treillis. Par ailleurs, toutes les chaînes (par exemple : $\mathbb{R}$, $\mathbb{N}$ et tous leurs sous-ensembles) sont des treillis où l'infimum et le supremum correspondent respectivement au minium et au maximum.

Un treillis $(L, \leq)$ peut également être vu comme la structure algébrique $(L, \wedge, \vee)$, où $\wedge$ et $\vee$ sont deux opérations binaires sur L définies par :

$$x \wedge y = \bigwedge\{x, y\} \quad \text{et} \quad x \vee y = \bigvee\{x, y\}.$$

Les deux opérations ainsi définies respectent les lois suivantes.

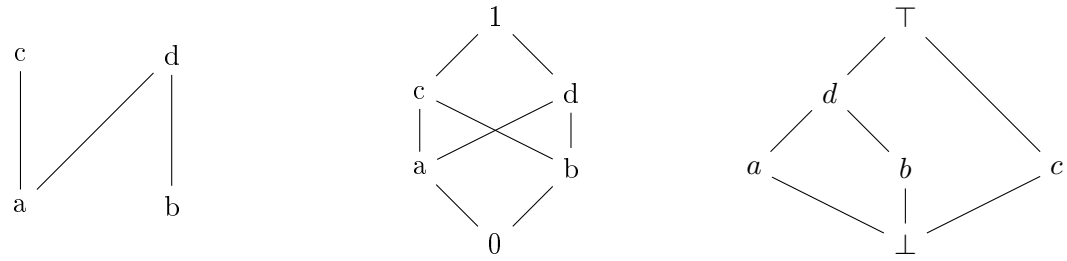


FIGURE 1.1 – Représentations d'ensembles ordonnés par des diagrammes de Hasse. La présence d'un lien entre deux éléments x, y indique que x et y sont comparables ; si $x \leq y$, alors x est représenté plus bas que y.

- Loi de *commutativité* : pour tout $x, y \in L$

$$x \wedge y = y \wedge x \quad \text{et} \quad x \vee y = y \vee x.$$

- Loi d'*associativité* : pour tout $x, y, z \in L$

$$x \wedge (y \wedge z) = (x \wedge y) \wedge z \quad \text{et} \quad x \vee (y \vee z) = (x \vee y) \vee z.$$

- Loi d'*absorption* : pour tout $x \in L$

$$x \vee (x \wedge y) = x \wedge (x \vee y) = x.$$

Le respect de ces trois lois implique, pour tout $x, y \in L$,

$$\begin{aligned} x \leq y \quad &\iff \quad x \wedge y = x, \\ x \leq y \quad &\iff \quad y \vee x = y. \end{aligned}$$

Pour tout $x_1, \ldots, x_n \in L$ (n étant un entier positif), on a

$$\bigwedge\{x_1, \ldots, x_n\} = x_1 \wedge \cdots \wedge x_n \quad \text{et} \quad \bigvee\{x_1, \ldots, x_n\} = x_1 \vee \cdots \vee x_n.$$

Enfin, notez que, pour tout $x_1, \ldots, x_n, y \in L$, on a les équivalences suivantes :

$$\begin{aligned} \left[\ \forall i \in \{1, \ldots, n\},\ x_i \geq a\ \right] \quad &\iff \quad x_1 \wedge \cdots \wedge x_n \geq a, \\ \left[\ \forall i \in \{1, \ldots, n\},\ x_i \leq a\ \right] \quad &\iff \quad x_1 \vee \cdots \vee x_n \leq a, \end{aligned}$$

ainsi que les implications suivantes :

$$\begin{aligned} \left[\ \exists i \in \{1, \ldots, n\},\ x_i \geq a\ \right] \quad &\implies \quad x_1 \vee \cdots \vee x_n \geq a, \\ \left[\ \exists i \in \{1, \ldots, n\},\ x_i \leq a\ \right] \quad &\implies \quad x_1 \wedge \cdots \wedge x_n \leq a. \end{aligned}$$

Ces équivalences et implications seront utilisées (implicitement) lors des démonstrations des deux chapitres suivants.

On dit qu'un treillis L est borné s'il contient une borne supérieure et une borne inférieure. On dénote ces bornes respectivement par $\bot_L$ et $\top_L$. Lorsque cela ne génère pas d'ambiguité, les bornes de L seront simplement dénotées par $\bot$ et $\top$.

Sous-treillis

Soit deux treillis $(L_1, \wedge_1, \vee_1)$ et $(L_2, \wedge_2, \vee_2)$. On dit que $(L_1, \wedge_1, \vee_1)$ est un sous-treillis de $(L_2, \wedge_2, \vee_2)$ si $L_1 \subseteq L_2$ et pour tout $x, y \in L_1$

$$x \wedge_1 y = x \wedge_2 y \quad \text{et} \quad x \vee_1 y = x \vee_2 y.$$

1.3 Fonctions sur les treillis

1.3.1 Notions générales

Soient f et g deux fonctions de X vers L, où L est un treillis. On notera $f \wedge g$ et $f \vee g$ les fonctions définies par

$$(f \wedge g)(\mathbf{x}) = f(\mathbf{x}) \wedge g(\mathbf{x}) \quad \text{et} \quad (f \vee g)(\mathbf{x}) = f(\mathbf{x}) \vee g(\mathbf{x}).$$

Une fonction $f : X \to Y$ est *injective* si, pour tout $x, y \in X$,

$$x \neq y \quad \Longrightarrow \quad f(x) \neq f(y).$$

Elle est *surjective*, si pour tout $y \in Y$, il existe $x \in X$ tel que $f(x) = y$. Elle est dite *bijective* si elle est injective et surjective.

1.3.2 Homomorphismes, plongements et isomorphismes

Soit deux posets $(X_1, \leq_1)$ et $(X_2, \leq_2)$. Un *homomorphisme d'ordre* de X_1 à X_2 est une fonction $h : X_1 \to X_2$ telle que, pour tout $x, y \in X_1$,

$$x \leq_1 y \quad \Longrightarrow \quad h(x) \leq_2 h(y).$$

La fonction h est un *isomorphisme d'ordre* de X_1 à X_2 si elle est bijective et si, pour tout $x, y \in X_1$,

$$x \leq_1 y \quad \Longleftrightarrow \quad h(x) \leq_2 h(y).$$

Soient deux treillis $(L_1, \wedge_1, \vee_1)$ et $(L_2, \wedge_2, \vee_2)$ et une fonction $h : L_1 \to L_2$. On dit que h est un *homomorphisme de treillis* de L_1 à L_2 si, pour tout $x, y \in L_1$

$$h(x \wedge_1 y) = h(x) \wedge_2 h(y) \quad \text{et} \quad h(x \vee_1 y) = h(x) \vee_2 h(y).$$

On dit que h est un *isomorphisme de treillis* de L_1 à L_2 si elle est bijective. La Figure 1.2 illustre la notion d'homomorphisme de treillis.

Notez qu'un homomorphisme d'ordre de L_1 à L_2 n'est pas nécessairement un homomorphisme de treillis. En revanche, un isomorphisme d'ordre de L_1 à L_2 est nécessairement un isomorphisme de treillis. On dit que deux posets sont isomorphes s'il existe un isomorphisme de l'un à l'autre.

Enfin, un *plongement* de L_1 dans L_2 est un homomorphisme injectif de L_1 à L_2. Notez que tout plongement de L_1 dans L_2 est un isomorphisme de L_1 à un sous-ensemble de L_2.

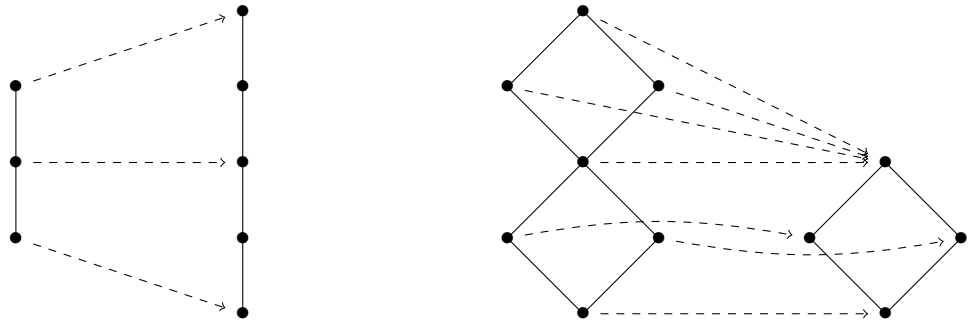


FIGURE 1.2 – Exemples d'homomorphismes entre treillis. Chaque élément du treillis d'origine est lié à son image par une flèche pointillée. La fonction représentée à gauche est un plongement, contrairement, à celle représentée à droite.

1.4 Treillis spéciaux

1.4.1 Treillis distributifs

Un treillis $(L, \wedge, \vee)$ est *distributif* si pour tout $x, y \in L$

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \quad \text{et} \quad x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

Remarque 2. Deux exemples typiques de treillis ne respectant pas la loi de distributivité sont les treillis M_3 et N_5 (voir Figure 1.3). Dans le treillis M_3, on a

$$a \vee (b \wedge c) = a,$$
$$(a \vee b) \wedge (a \vee c) = \top.$$

Dans le treillis N_5, on a

$$a \vee (b \wedge c) = a,$$
$$(a \vee b) \wedge (a \vee c) = b.$$

En fait, un treillis est distributif si et seulement s'il ne possède aucun sous-treillis de la forme M_3 ou N_5 (voir [59], Chapitre II, Théorème 1).

FIGURE 1.3 – Les treillis M_3 (gauche) et N_5 (droite).

1.4.2 Treillis Booléen

Soit un treillis borné L. Pour tout $a, b \in L$, a est un complément de b si $a \wedge b = \bot$ et $a \vee b = \top$. Un treillis est *Booléen* si il est distributif et borné, et si chaque élément du treillis possède un unique complément.

1.4.3 Treillis d'ensembles

On appelle *treillis d'ensembles* tout treillis $(\mathbf{X}, \cap, \cup)$, où $\mathbf{X}$ est un ensemble d'ensembles, et où $\cap$ et $\cup$ correspondent respectivement à l'opération d'intersection et d'union. La Figure 1.4 fournit quelques exemples de treillis d'ensembles. Dans un tel treillis, la relation d'ordre correspond à

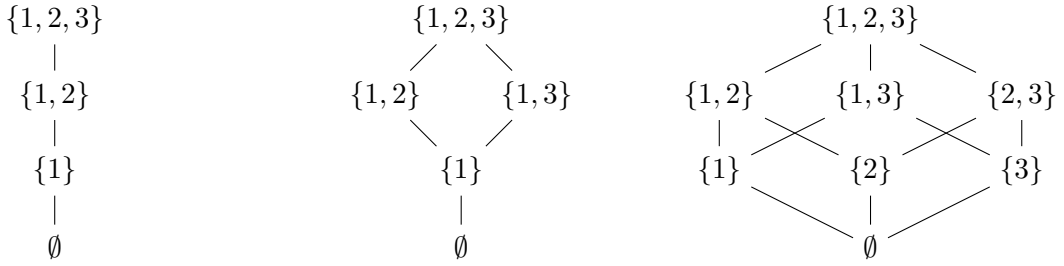


FIGURE 1.4 – Exemples de treillis d'ensembles

la relation d'inclusion des ensembles. Un cas particulier de treillis d'ensembles est le treillis de l'*ensemble des parties* d'un ensemble. Soit X un ensemble : l'ensemble des parties de X, dénoté par 2^X, est l'ensemble des sous-ensembles de X. Le treillis de l'ensemble des parties de X est l'ensemble des sous-ensembles de X ordonnés par inclusion $(2^X, \subseteq)$. Tout treillis d'ensemble dont l'élément maximal est un ensemble fini X est un sous-treillis de $(2^X, \subseteq)$.

Les treillis d'ensembles sont tous distributifs. De plus, un treillis fini est distributif si et seulement si il est isomorphe à un treillis d'ensembles. Pour tout ensemble X, $(2^X, \subseteq)$ est un treillis Booléen. De plus, tout treillis fini est Booléen si et seulement si il est isomorphe au treillis de l'ensemble des parties d'un ensemble.

1.5 Plongements et représentations des treillis distributifs

1.5.1 Join-irréductibilité, meet-irréductibilité

Un element x de L est dit *join-irréductible* si $x \neq \bot$ (dans le cas où L contient une borne inférieure) et si pour tout $a, b \in L$

$$a \vee b = x \quad \Longrightarrow \quad \big[x = a \text{ ou } x = b\big].$$

Autrement dit, un élément qui n'est pas join-irréductible est soit la borne inférieure de L, soit un élément de L pouvant être obtenu par le supremum de deux autres éléments de L. De manière analogue, on dit que x est *meet-irréductible* si $x \neq \top$ (dans le cas où L contient une borne supérieure) et si pour tout $a, b \in L$

$$a \wedge b = x \quad \Longrightarrow \quad \big[x = a \text{ ou } x = b\big].$$

On dénotera par $\mathcal{J}(L)$ l'ensemble des éléments join-irréductibles de L. L'ensemble $\mathcal{J}(L)$ est impliqué dans les méthodes de représentation des treillis distributifs que nous présentons dans les deux sous-sections suivantes. Pour des raisons de simplicité, les représentations duales faisant appel à l'ensemble des éléments meet-irréductibles du treillis ne sont pas traitées.

1.5.2 Représentation de Birkhoff

Il existe une correspondance entre les posets finis et les treillis distributifs finis. Plus précisément, tout treillis distributif est caractérisé par le poset de ses éléments join-irréductibles et, à l'inverse, tout ordre partiel donne naissance à un treillis.

Voyons d'abord comment générer un treillis à partir de chaque poset fini. Pour tout poset fini X, soit $\mathcal{O}(X)$ l'ensemble défini par

$$\mathcal{O}(X) = \Big\{ \bigcup_{x \in A} \downarrow x \;\Big|\; A \subseteq X \Big\}.$$

La structure $(\mathcal{O}(X), \cap, \cup)$ est un treillis, puisque pour tout $A, B \in \mathcal{O}(X)$, $A \cap B$ et $A \cup B$ appartiennent à $\mathcal{O}(X)$. De plus l'ensemble des éléments join-irreductibles de ce treillis est

$$\mathcal{J}(\mathcal{O}(X)) = \{\downarrow x \mid x \in X\}.$$

L'ensemble $\mathcal{J}(\mathcal{O}(X))$, ordonné par la relation d'inclusion $\subseteq$, est isomorphe à X. En effet, pour tout $x, y \in X$, on a $x \leq y$ si et seulement si $\downarrow x \subseteq \downarrow y$.

La correspondance entre les treillis et les posets est établie par le théorème suivant, qui décrit de quelle manière un treillis isomorphe à L peut être construit à partir du poset $\mathcal{J}(L)$ des éléments join-irréductibles de L.

Théorème de représentation de Birkhoff. Soit un treillis distributif fini L. Le théorème de représentation de Birkhoff établit que la fonction $h : L \to \mathcal{O}(\mathcal{J}(L))$ définie par

$$h(x) = \big\{ y \in \mathcal{J}(L) \mid y \leq x \big\}$$

est un isomorphisme de L vers $\mathcal{O}(\mathcal{J}(L))$.

Notez que h est également un plongement de L dans le treillis $2^{\mathcal{J}(L)}$. Le théorème de représentation de Birkhoff permet de voir L comme un treillis d'ensembles dans lequel chaque élément de L est représenté par l'ensemble des éléments join-irréductibles inférieurs ou égaux à lui-même.

Soit k la taille de $\mathcal{J}(L)$; on dénote les éléments de $\mathcal{J}(L)$ par $a_1, \ldots, a_k$. Le treillis $2^{\mathcal{J}(L)}$ et le treillis Booléen $\{0,1\}^k$ (dont les éléments sont des tuples de valeurs binaires) sont isomorphes. Un isomorphisme intuitif entre ces deux treillis est la fonction $h' : 2^{\mathcal{J}(L)} \to \{0,1\}^k$ définie par

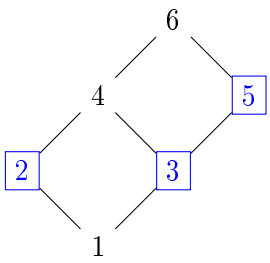
$$h'(X) = (b_1, \ldots, b_k), \quad \text{où} \quad \forall i \in [k], \begin{cases} b_i = 1 \text{ si } a_i \in X \\ b_i = 0 \text{ sinon.} \end{cases}$$

Par conséquent, L peut également être plongé dans $\{0,1\}^k$ (autrement dit, les éléments de L peuvent être représentés par des tuples binaires). La figure 1.5 illustre le plongement d'un treillis L dans $2^{\mathcal{J}(L)}$ et dans $\{0,1\}^k$.

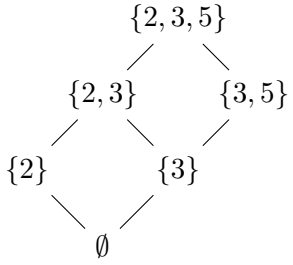
1.5.3 Plongement de Dilworth

Soit un treillis distributif fini L. Le théorème de représentation de Birkhoff montre que L peut être vu comme un sous-treillis du treillis $\{0,1\}^{|\mathcal{J}(L)|}$. De manière plus générale, L peut être vu comme un sous-treillis d'un produit de chaines [89]. Pour tout entier positif k, on dénote par $[k]$ l'ensemble $\{1, \ldots, k\}$. Soit un entier positif k et des chaines $L_1, \ldots, L_k$ telles que

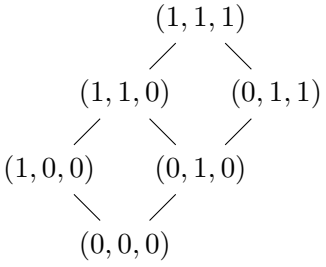
$$L_1 \cup \cdots \cup L_k = \mathcal{J}(L) \cup \{\bot\} \quad \text{et} \quad \forall i, j \in [k],\ \big[i \neq j\big] \Rightarrow \big[L_i \cap L_j = \{\bot\}\big].$$



Un treillis distributif L.



Le treillis $\mathcal{O}(\mathcal{J}(L))$.



Un sous-treillis de $\{0,1\}^{|\mathcal{J}(L)|}$, isomorphe à L et $\mathcal{O}(\mathcal{J}(L))$.

FIGURE 1.5 – Trois treillis distributifs isomorphes.

Le *plongement de Dilworth* de L dans $L_1 \times \cdots \times L_k$ est la fonction $\gamma : L \to L_1 \times \cdots \times L_k$ définie par

$$\gamma(x) = \big(x|_1, \ldots, x|_k\big),$$

où, pour tout $x \in L$ et $i \in [k]$,

$$x|_i = \bigvee \Big\{ y \in L_i \;\Big|\; y \leq x \Big\}.$$

Remarque 3. La fonction γ est bien un homomorphisme puisque pour tout $x, y \in L$, si $x \leq x'$ alors pour tout $i \in [n]$ on a $x|_i \leq x'|_i$. De plus, elle associe à $\bot$ et $\top$ l'élément minimal et l'élément maximal de $L_1 \times \cdots \times L_k$, respectivement. Enfin, puisque tout élément d'un treillis est égal au supremum des join-irréductibles qui sont inférieurs à lui, on a

$$x = \bigvee \{ y \in \mathcal{J}(L) \mid y \leq x \} = \bigvee_{i \in [k]} \bigvee \{ y \in L_i \mid y \leq x \} = c_1^x \vee \cdots \vee c_k^x.$$

Cette égalité permet de vérifier que γ est bijective.

La fonction γ étant un plongement, on a

$$a \leq b \iff \gamma(a) \leq \gamma(b) \iff \forall i \in [k],\ a|_i \leq b|_i.$$

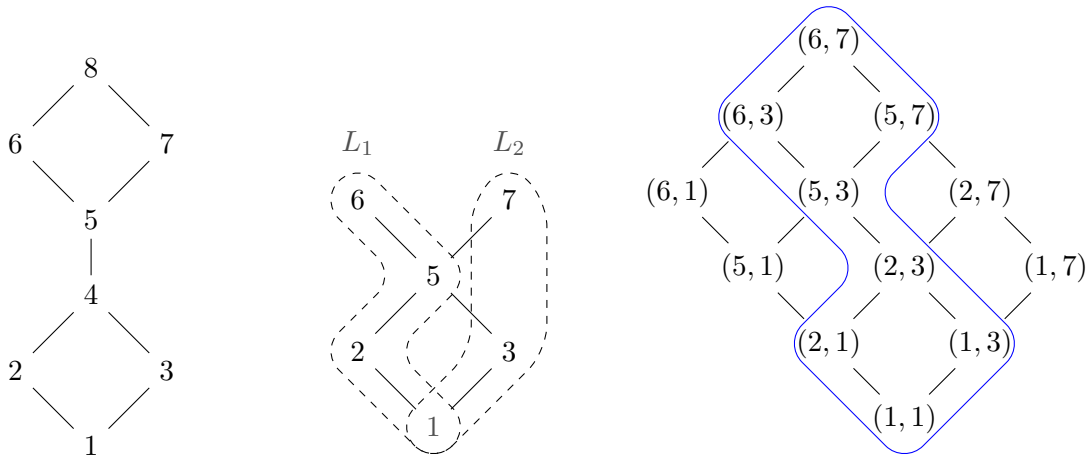
ainsi que

$$\gamma(a \wedge b) = \big(a|_1 \wedge b|_1, \ldots, a|_k \wedge b|_k\big) \quad \text{et} \quad \gamma(a \vee b) = \big(a|_1 \vee b|_1, \ldots, a|_k \vee b|_k\big). \tag{1.1}$$

Pour tout treillis L, il est possible de définir au moins un plongement de Dilworth. Chaque plongement de Dilworth est déterminé par le choix des chaînes $L_1, \ldots, L_k$.

Remarque 4. Toute chaîne finie est isomorphe à un intervalle de nombres entiers. Par conséquent, le plongement de Dilworth de L dans $\prod_{i=1}^{k} L_i$ peut être vu comme une manière de représenter chaque élément de L par un tuple de valeurs entières de taille k. Remarquez également que le plongement de L dans $\{0,1\}$ permis par le théorème de représentation de Birkhoff est équivalent à un plongement de Dilworth de L dans $L_1 \times \cdots \times L_k$ où, pour tout $i \in [k]$, chaque chaîne L_i contient exactement 2 éléments : un join-irreductible de L et $\bot$. On dira d'un tel plongement de Dilworth qu'il est *trivial*.

Soit un poset X. Un sous-ensemble de X est dit *indépendant* si tous ses éléments sont incomparables.



Un treillis L.

L'ensemble ordonné $\mathcal{J}(L) \cup \{1\}$ des join-irréductibles de L.

Treillis $L_1 \times L_2$. L'image du plongement de L est entourée en bleu.

FIGURE 1.6 – Illustration d'un plongement de Dilworth.

Théorème de décomposition de Dilworth [38]. Soit un poset X ayant un sous-ensemble indépendant de taille k, et aucun sous-ensemble indépendant de taille $k+1$. Alors X est l'union de k chaînes.

La taille du plus grand sous-ensemble indépendant de X (qu'on appelle largeur de X, et qu'on dénote $w(X)$) donne le nombre minimal de chaînes nécessaires afin de partitionner un ensemble ordonné X.

On peut déterminer $w(X)$ en $O(|X|)$, et un partitionnement de X en w chaînes peut être effectué en $O(w(X)|X|^2)$ [64]. Pour un treillis L dont on connait l'ensemble $\mathcal{J}(L)$ des join-irreductibles, on peut donc déterminer un plongement de Dilworth optimal en

$$O\Big(w\big(\mathcal{J}(L)\big)\big|\mathcal{J}(L)\big|^2\Big).$$

1.6 Implémentation des treillis finis

La première partie de cette thèse est principalement dédiée à la description d'algorithmes faisant appel aux opérations binaires du treillis ($\wedge$ et $\vee$), ainsi qu'à la complexité de ces algorithmes. La complexité de $\wedge$ et $\vee$ dépend de la représentation du treillis. Ici nous présentons deux exemples de représentation possible qui seront utiles dans les chapitres suivants.

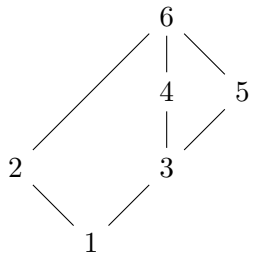
1.6.1 Représentation en tables

Une première méthode consiste à représenter chaque élément du treillis par un entier, et à représenter les opérations $\wedge$ et $\vee$ par des tableaux à deux dimensions.

On suppose que $L = \{1, \ldots, |L|\}$. On appelle $\mathtt{T}_\wedge$ le tableau à deux dimension de taille $|L| \times |L|$ dont les valeurs sont définies de la manière suivante : pour tout $a, b \in L$,

$$\mathtt{T}_\wedge[a][b] = a \wedge b.$$

On appelle $\mathtt{T}_\vee$ le tableau défini de manière analogue en fonction de $\vee$.



$\mathtt{T}_\wedge$	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	2	1	1	1	2
3	1	1	3	3	3	3
4	1	1	3	4	3	4
5	1	1	3	3	5	5
6	1	2	3	4	5	6

$\mathtt{T}_\vee$	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	2	6	6	6	6
3	3	6	3	4	5	6
4	4	6	4	4	6	6
5	5	6	5	6	5	6
6	6	6	6	6	6	6

FIGURE 1.7 – Exemple de représentation d'un treillis par des tableaux.

L'implémentation des opérations $\wedge$ et $\vee$ par des tableaux permet de retourner le supremum et l'infimum de n'importe quelle paire d'éléments en $O(1)$. De plus $a \leq b$ est équivalent à $a \wedge b = a$; puisque la comparaison d'entiers s'effectue en $O(1)$, cette implémentation permet également de tester la condition $a \leq b$ en $O(1)$.

Cette implémentation requiert de stocker en mémoire deux tableaux de taille $O(|L|^2)$. De plus, l'initialisation des deux tableaux à partir d'une implémentation différente du treillis se fait en $O(|L|^2\Delta)$, où $O(\Delta)$ est la complexité au pire cas des opérations $\wedge$ et $\vee$ dans l'implémentation originale du treillis. La représentation du treillis par des tableaux n'a donc d'intérêt que si le treillis est de taille raisonnable, et si on prévoit de faire appel à ses opérations de manière intensive.

1.6.2 Treillis distributifs : représentation à base de tuples d'entiers

Le plongement de Dilworth permet de représenter n'importe quel treillis distributif fini L comme un sous-treillis d'un produit de k chaînes. En regardant chacune de ces chaînes comme un intervalle entier, on peut représenter chaque élément de L par une séquence de k nombres entiers (typiquement implémentée sous forme de tableau). L'infimum de deux éléments $a, b \in L$ (représentés par deux tableaux d'entiers de taille k, respectivement $\mathtt{T}_a$ et $\mathtt{T}_b$) est représenté par le tableau $\mathtt{T}$ tel que

$$\forall i \in [k], \quad \mathtt{T}[i] = \mathtt{T}_a[i] \wedge \mathtt{T}_b[i].$$

Les valeurs de ce tableau sont calculées à partir de $\mathtt{T}_a$ et $\mathtt{T}_b$ en $O(k)$. Le supremum s'obtient de manière analogue par le maximum. Enfin, puisque $a \leq b$ si et seulement si $\mathtt{T}_a[i] \leq \mathtt{T}_b[i]$ pour tout $i \in [k]$, on peut tester la condition $a \leq k$ en $O(k)$. Notez qu'on a toujours $k \leq |\mathcal{J}(L)|$. Si l'on se base sur un plongement de Dilworth trivial, alors $k = |\mathcal{J}(L)|$.

Chapitre 2

Fonctions latticielles polynomiales : cas général

2.1 Introduction

Dans le chapitre précédent, nous avons défini la notion de treillis ainsi que les opérations binaires d'infimum et de supremum, respectivement dénotées par les opérateurs $\wedge$ et $\vee$. On s'intéresse maintenant aux fonctions qui peuvent être exprimées à partir de ces opérateurs, qu'on appelle *fonctions latticielles polynomiales* (FLP). Ce chapitre présente la définition des FLP et introduit le *problème d'interpolation*, qui est celui de déterminer s'il existe au moins une FLP compatible avec un ensemble d'exemples (représentés par une fonction partielle) et, lorsque que cela est le cas, de retourner l'une de ces FLP. Ce problème sera traité dans ce chapitre et le suivant. Les contributions majeurs présentées dans ce chapitre sont les suivantes :

- une caractérisation des fonctions partielles qui sont interpolées par au moins une FLP,
- une méthode permettant de calculer la représentation d'une FLP interpolant une fonction $f : D \to L$ (où $D \subseteq L^n$) si une telle FLP existe, en temps polynomial par rapport à n, à la taille de D et à la taille de L.

On utilisera les notations et conventions suivantes : n sera toujours un entier positif, et L un treillis. De plus, L^n dénotera le produit Cartésien du treillis L répété n fois, et D dénotera un sous-ensemble fini de L^n. Pour tout entier positif n, on dénotera l'ensemble $\{1, \ldots, n\}$ par $[n]$. Les tuples appartenant à L^n seront dénotés par un symbole gras (typiquement $\mathbf{x}$). Les composantes de $\mathbf{x} \in L^n$ seront dénotées par $x_1, \ldots, x_n$. Les FLP seront dénotés par p, q, r, ...

2.2 Fonctions latticielles polynomiales

Une *fonction latticielle polynomiale* (FLP) est une fonction pouvant être exprimée à partir de variables, de constantes, et des opérateurs $\wedge$ et $\vee$. Formellement, on définit l'ensemble des FLP de L^n à L de la manière récursive suivante [76].

1. Pour tout $i \in [n]$, la projection $p : L^n \to L$ définie par

$$p(x_1, \ldots, x_n) \mapsto x_i$$

est une FLP de L^n à L.

2. Pour tout $c \in L$, la fonction constante $p : L^n \to L$ définie par

$$p(x_1, \ldots, x_n) = c$$

est une FLP de L^n à L.

3. Si p et q sont des FLP de L^n à L, alors $p \wedge q$ et $p \vee q$ sont des FLP de L^n à L.

On appelle $\mathcal{P}_L^{(n)}$ l'ensemble des FLP de L^n à L. Notez que, dans le cas où L est fini, l'ensemble $\mathcal{P}_L^{(n)}$ est également fini (puisque $\mathcal{P}_L^{(n)}$ est inclus dans l'ensemble des fonctions de L^n vers L).

Exemple 5. Considérons la chaîne $L = \{1, 2, 3\}$. Les fonctions $p, q, r, s, t : L^2 \to L$, définies par les identités ci-dessous, appartiennent à $\mathcal{P}_L^{(2)}$:

$$\begin{aligned}
p(x_1, x_2) &= 2, && \text{(règle 2)}\\
q(x_1, x_2) &= x_1, && \text{(règle 1)}\\
r(x_1, x_2) &= x_2, && \text{(règle 1)}\\
s(x_1, x_2) &= p(x_1, x_2) \vee r(x_1, x_2) = 2 \vee x_2, && \text{(règle 3)}\\
t(x_1, x_2) &= q(x_1, x_2) \wedge s(x_1, x_2) = x_1 \wedge (2 \vee x_2) && \text{(règle 3).}
\end{aligned}$$

On utilise les parenthèses pour moduler la priorité des opérateurs, lorsque nécessaire. Par défaut, l'opérateur $\wedge$ est prioritaire sur l'opérateur $\vee$. Enfin, puisque $\wedge$ et $\vee$ sont des opérateurs binaires, les LFP se représentent naturellement sous forme d'arbres binaires, où chaque nœud non feuille est l'un des opérateurs $\wedge$ et $\vee$ et chaque feuille est soit une variable soit une constante. Voir, par exemple, Figure 2.1.

Remarque 6. Les fonctions booléennes non décroissantes [32] sont les fonctions de $\{0,1\}^n$ à $\{0,1\}$ qui peuvent être exprimées à l'aide de variables et des opérateurs OR et AND (sans l'opérateur de négation). Puisque ($\{0,1\}$, AND, OR) est un treillis dont AND est l'infimum et dont OR est le supremum, les fonctions booléennes non décroissantes sont des FLP, et les FLP peuvent être vues comme une généralisation des fonctions booléennes non décroissantes sur des échelles plus riches que $\{0,1\}$.

Remarque 7. Les termes *lattice polynomial* ou *polynomial* étant souvent utilisés pour désigner les expressions formées à partir des opérateurs $\wedge$ et $\vee$, et de variables (sans constantes) [10, 59], les FLP sont parfois appelées *weighted lattice polynomial functions* (voir [76]) afin de les différencier des fonctions exprimées par des polynômes sans constantes. Enfin, les FLP sont parfois simplement appelées *lattice functions* [51].

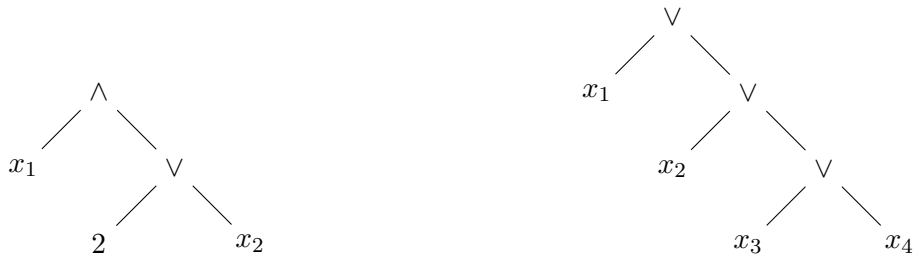


FIGURE 2.1 – Représentations en arbres des FLP définies par $p(x_1, x_2) = x_1 \wedge (2 \vee x_2)$ (à gauche) et $q(x_1, \ldots, x_4) = x_1 \vee x_2 \vee x_3 \vee x_4$ (à droite).

2.3 Problème d'interpolation

Le *problème d'interpolation* est celui de trouver une FLP cohérente avec un ensemble d'exemples de la forme (arguments, résultat). Cet ensemble d'exemples est représenté par une fonction partielle. Commençons par introduire quelques définitions.

Soit un treillis L, un ensemble $D \subseteq L^n$, une fonction partielle $f : D \to L$ et une FLP $p \in \mathcal{P}_L^{(n)}$. On dit que p et f coïncident sur $\mathbf{x} \in D$ si $p(\mathbf{x}) = f(\mathbf{x})$, et qu'ils coïncident sur $X \subseteq D$ si l'égalité $p(\mathbf{x}) = f(\mathbf{x})$ est vérifiée pour tout $\mathbf{x} \in X$. Lorsque p et f coïncident sur D, on dit que p est une interpolation de f, que p interpole f, ou encore que p est une solution au problème d'interpolation pour f. On appelle $\mathcal{I}(f)$ l'ensemble des FLP qui interpolent f. Si cet ensemble est non vide, on dit que f est une *FLP partielle*.

Étant donné une fonction $f : D \to L$, nous nous intéressons aux trois tâches suivantes.

- Décider si f est une FLP partielle. Autrement dit, décider s'il existe au moins une FLP interpolant f.
- Retourner une interpolation de f, s'il en existe (c'est à dire un élément quelconque de $\mathcal{I}(f)$, si $\mathcal{I}(f) \neq \emptyset$).
- Retourner une description satisfaisante de $\mathcal{I}(f)$.

Le troisième point est volontairement formulé de manière informelle, et sera abordé à nouveau à la fin du chapitre.

Dans ce chapitre et le suivant, nous traitons les deux premières tâches du problème d'interpolation d'un point de vue algorithmique. Plus précisément, nous décrivons une méthode qui réalise ces deux premières tâches en temps polynomial par rapport à l'arité de f, la taille de son domaine, et la taille du treillis sur lequel elle est définie.

Le problème d'interpolation n'a, à notre connaissance, pas été traité sous cet angle par le passé. Les travaux qui abordent cette question (voir [4] dans le cadre plus général de l'algèbre universelle, ou [24], dans le cadre des treillis distributifs) visent plutôt à caractériser les différentes composantes du problème, à savoir l'ensemble des FLP partielles ou l'ensemble $\mathcal{I}(f)$, qu'à proposer des algorithmes efficaces pour calculer les objets ainsi caractérisés.

La méthode présentée dans ce chapitre repose sur une caractérisation des FLP partielles présentée dans la Section 2.5.

2.4 Congruences, tolerances and comprefs

Une des propriétés des FLP est de préserver certains types de relations binaires sur les treillis (les *congruences* et, plus généralement, les *tolérances*). Dans cette section nous rappelons la définition de ces relations et leurs liens connus avec les FLP. Nous montrons ensuite que, dans l'optique de caractériser les FLP partielles, il est préférable de considérer un type de relation plus général, qu'on appellera *compref*.

Soit un treillis L. Une relation R sur L est *compatible* si, pour tout $a, b, c, d \in L$

$$(a, b) \in R \text{ et } (c, d) \in R \quad \Longrightarrow \quad (a \wedge c, b \wedge d) \in R \text{ et } (a \vee c, b \vee d) \in R.$$

Remarque 8. Une relation compatible sur L peut être vue comme un sous-ensemble de L^2 fermé par les opérations $\wedge$ et $\vee$ sur L^2, définies par

$$(a_1, a_2) \wedge (b_1, b_2) = (a_1 \wedge b_1, a_2 \wedge b_2) \quad \text{et} \quad (a_1, a_2) \vee (b_1, b_2) = (a_1 \vee b_1, a_2 \vee b_2)$$

pour tout $(a_1, a_2), (b_1, b_2) \in L^2$.

Une *tolérance* sur L est une relation compatible, symétrique et reflexive sur L. Une *congruence* sur L est une tolérance transitive, c'est à dire une relation d'équivalence compatible sur L. Notez que la notion de congruence est centrale en théorie des treillis [36], et plus généralement de l'algèbre universelle [17], car elle est étroitement liée à la notion d'homomorphisme. La notion de tolérance, quant à elle, apparait dans des contextes plus spécifiques, notamment dans la caractérisation des FLP [5].

Soit $f : D \to L$ (où $D \subseteq L^n$). Pour toute relation R sur L, on dit que f préserve R si elle vérifie

$$\forall \mathbf{x}, \mathbf{x'} \in D, \quad \big[\forall i \in [n],\ (x_i, x'_i) \in R\big] \Rightarrow \big[(f(\mathbf{x}), f(\mathbf{x'})) \in R\big]. \tag{2.1}$$

Si (2.1) est vérifiée pour toute congruence R, on dit que f préserve les congruences. Si (2.1) est vérifiée pour toute tolérance R, on dit que f préserve les tolérances. Notez que, puisque toute congruence est une tolérance, les fonctions qui préservent les tolérances préservent également les congruences.

Les FLP préservent les congruences [65], et correspondent exactement aux fonctions non décroissantes qui préservent les tolérances [5, 69]. Cependant, il existe des fonctions partielles non décroissantes qui préservent les tolérances et qui ne sont pas des FLP partielles, comme l'illustre l'exemple suivant.

Exemple 9. Soit L le treillis représenté dans la Figure 2.2. La fonction partielle unaire $f : \{a, b\} \to L$, définie par $f(a) = b$ et $f(b) = a$, préserve les tolérances. En effet, puisque toute tolérance R sur L est une relation symétrique,

$$\text{si} \quad (a, b) \in R \quad \text{alors} \quad (f(a), f(b)) = (b, a) \in R.$$

Cependant, on peut facilement se convaincre qu'aucune FLP n'interpole f (une vérification plus formelle peut être faite en utilisant le Théorème 15 de la section suivante).

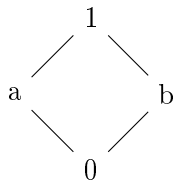


FIGURE 2.2 – Un treillis.

Cet exemple suggère que les notions de congruence et de tolérance, bien qu'étroitement liées à celle de FLP, ne sont pas des outils adaptés pour caractériser les FLP partielles. Pour cela, on affaiblit la notion de tolérance.

Définition 10. Un *compref* sur L est une relation réflexive et compatible sur L.

Remarque 11. Un compref est une tolérance si et seulement si il est symétrique.

Dans la suite, le symbole utilisé par défaut pour dénoter un compref sera Я. On dit qu'une fonction préserve les comprefs si elle vérifie (2.1) pour tout compref. Comme nous allons le montrer dans la section suivante, les fonctions partielles de L^n à L qui préservent les comprefs correspondent exactement aux FLP partielles n-aires sur L. L'intérêt de cette caractérisation est la suivante : si l'on sait identifier efficacement les fonctions qui préservent les comprefs, alors on peut identifier les fonctions partielles qui peuvent être interpolées par une FLP.

Notez que le nombre maximal de comprefs sur un treillis fini augmente exponentiellement avec la taille du treillis [1]. Cependant il est possible d'identifier les fonctions qui préservent les comprefs sans énumérer tous les comprefs sur L.

Définition 12. Soit $\mathbf{x}, \mathbf{x'} \in L^n$. On appelle $Я_{\mathbf{x},\mathbf{x'}}$ l'intersection de tous les comprefs Я sur L qui vérifient

$$\forall i \in [n],\ (x_i, x'_i) \in Я. \tag{2.2}$$

Remarque 13. Soit $D \subseteq L^n$. Une fonction $f : D \to L$ préserve les comprefs si et seulement si

$$\forall \mathbf{x}, \mathbf{x'} \in D, \quad (f(\mathbf{x}), f(\mathbf{x'})) \in Я_{\mathbf{x},\mathbf{x'}}.$$

Proposition 14. La relation $Я_{\mathbf{x},\mathbf{x}'}$ est le plus petit compref sur L qui vérifie (2.2).

Démonstration. Pour tout compref Я sur L qui vérifie (2.2), on a $Я_{\mathbf{x},\mathbf{x}'} \subseteq Я$. Il reste donc à prouver que $Я_{\mathbf{x},\mathbf{x}'}$ est un compref. Puisque la relation $Я_{\mathbf{x},\mathbf{x}'}$ est définie comme l'intersection de relations reflexives sur L, elle est réflexive. Pour prouver que $Я_{\mathbf{x},\mathbf{x}'}$ est compatible, on considère $(a, b), (c, d) \in Я_{\mathbf{x},\mathbf{x}'}$. Pour tout compref Я vérifiant (2.2), on a $Я_{\mathbf{x},\mathbf{x}'} \subseteq Я$ et donc $(a, b), (c, d) \in Я$ et, puisque les comprefs sont compatibles,

$$(a \vee c, b \vee d), (a \wedge c, b \wedge d) \in Я.$$

Donc, par définition, $Я_{\mathbf{x},\mathbf{x}'}$ contient également $(a \vee c, b \vee d)$ et $(a \wedge c, b \wedge d)$. La relation $Я_{\mathbf{x},\mathbf{x}'}$ est donc bien compatible. □

En résumé : pour déterminer si $f : D \to L$ préserve les comprefs, il suffit de tester si $(f(\mathbf{x}), f(\mathbf{x'})) \in Я_{\mathbf{x},\mathbf{x}'}$ pour tout $\mathbf{x}, \mathbf{x'} \in D$. Or, puisque $Я_{\mathbf{x},\mathbf{x}'}$ est la plus petite relation réflexive et compatible vérifiant (2.2), elle est égale à la fermeture de l'ensemble

$$\big\{(x_i, x'_i) \;\big|\; i \in [n]\big\} \cup \big\{(a, a) \;\big|\; a \in L\big\}$$

par les opérations $\wedge$ et $\vee$ (considérées sur le treillis L^2).

2.5 Caractérisation des FLP partielles

Nous allons maintenant utiliser les notions de la section précédente pour caractériser les FLP partielles. Plus précisément nous allons prouver le théorème suivant.

Théorème 15. Soit un treillis L, $D \subseteq L^n$, et $f : D \to L$. Les conditions suivantes sont équivalentes.

1. f est une FLP partielle,
2. f préserve les comprefs,
3. pour tout $\mathbf{x}, \mathbf{x'} \in D$, on a $(f(\mathbf{x}), f(\mathbf{x'})) \in Я_{\mathbf{x},\mathbf{x}'}$.

Notez que les conditions 2 et 3 sont équivalentes par définition de $Я_{\mathbf{x},\mathbf{x}'}$. Ce théorème synthétise deux faits importants : premièrement, toute fonction $f : D \to L$ peut être interpolée par une FLP si et seulement si il existe une FLP qui coïncide avec f sur chaque paire d'éléments de D. Deuxièmement, on peut vérifier s'il existe une FLP coïncidant avec f sur $\mathbf{x}, \mathbf{x'} \in D$ en construisant l'ensemble $Я_{\mathbf{x},\mathbf{x}'}$.

On commence par le lemme suivant.

1. toute congruence est un compref, et le nombre de congruences sur une chaîne L correspond au nombre de partitions de L en intervalles, c'est à dire $2^{|L|-1}$.

Lemme 16. Soit $\mathbf{x}, \mathbf{x'} \in D$. Il existe $p \in \mathcal{P}_L^{(n)}$ tel que $p(\mathbf{x}) = f(\mathbf{x})$ et $p(\mathbf{x'}) = f(\mathbf{x'})$ si et seulement si $(f(\mathbf{x}), f(\mathbf{x'})) \in \text{Я}_{\mathbf{x},\mathbf{x'}}$.

Démonstration. Soit R la relation définie par

$$R = \Big\{ \big(p(\mathbf{x}), p(\mathbf{x'})\big) \;\Big|\; p \in \mathcal{P}_L^{(n)} \Big\}.$$

Clairement, il existe $p \in \mathcal{P}_L^{(n)}$ telle que $p(\mathbf{x}) = f(\mathbf{x})$ et $p(\mathbf{x'}) = f(\mathbf{x'})$ si et seulement si $(f(\mathbf{x}), f(\mathbf{x'})) \in R$. Or, la définition des FLP nous permet de déduire que

- pour tout $i \in [n]$ on a $(x_i, x'_i) \in R$, puisque $\mathcal{P}_L^{(n)}$ contient toutes les projections de L^n à L (règle 1).
- pour tout $a \in L$ on $(a, a) \in R$, puisque $\mathcal{P}_L^{(n)}$ contient toutes les fonctions constantes de L^n à L (règle 2),

et enfin que R est la fermeture de $\big\{(x_i, x'_i) \;\big|\; i \in [n]\big\} \cup \big\{(a, a) \;\big|\; a \in L\big\}$ par les opérations $\wedge$ et $\vee$ (règle 3). On a donc $R = \text{Я}_{\mathbf{x},\mathbf{x'}}$. □

Corollaire 17. Toute FLP partielle préserve les comprefs.

Lemme 18. Toute fonction qui préserve les comprefs est une FLP partielle.

Démonstration. Soit $D = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\} \subseteq L^n$ et $f : D \to L$ une fonction préservant les comprefs. Pour tout $i, j \in [m]$ on a $\big(f(\mathbf{x}^i), f(\mathbf{x}^j)\big) \in \text{Я}_{\mathbf{x}^i,\mathbf{x}^j}$. Donc, d'après le Lemme 16, pour chaque $i, j \in [m]$, il existe une FLP, qu'on dénotera $p^{i,j}$, qui coïncide avec f sur $\{\mathbf{x}^i, \mathbf{x}^j\}$, c'est à dire telle que

$$p^{i,j}(\mathbf{x}^i) = f(\mathbf{x}^i) \quad \text{et} \quad p^{i,j}(\mathbf{x}^j) = f(\mathbf{x}^j).$$

Pour tout $i \in [m]$, on pose

$$p^i = p^{i,1} \wedge \cdots \wedge p^{i,m}.$$

Puisque

$$\forall i \in [m], \quad p^{i,1}(\mathbf{x}^i) = \cdots = p^{i,m}(\mathbf{x}^i) = f(\mathbf{x}^i),$$

on a

$$\forall i \in [m], \quad p^i(\mathbf{x}^i) = f(\mathbf{x}^i). \tag{2.3}$$

De plus, puisque

$$\forall i, j \in [m], \quad \Big[p^i(\mathbf{x}^j) = p^{i,1}(\mathbf{x}^j) \wedge \cdots \wedge p^{i,m}(\mathbf{x}^j) \quad \text{et} \quad p^{i,j}(\mathbf{x}^j) = f(\mathbf{x}^j)\Big]$$

on a

$$\forall i, j \in [m], \quad p^i(\mathbf{x}^j) \leq f(\mathbf{x}^j). \tag{2.4}$$

Soit

$$p = p^1 \vee \cdots \vee p^m.$$

Puisque

$$\forall i \in [m], \quad p(\mathbf{x}^i) = p^1(\mathbf{x}^i) \vee \cdots \vee p^m(\mathbf{x}^i),$$

il suit de (2.3) que

$$\forall i \in [m], \quad p(\mathbf{x}^i) \geq f(\mathbf{x}^i),$$

et il suit de (2.4) que

$$\forall i \in [m], \quad p(\mathbf{x}^i) \leq f(\mathbf{x}^i).$$

On vient de prouver que p est une interpolation de f. De manière duale, on peut montrer que la FLP

$$(p^{1,1} \vee \cdots \vee p^{1,m}) \wedge \cdots \wedge (p^{m,1} \vee \cdots \vee p^{m,m})$$

est également une interpolation de f. Donc toute fonction qui préserve les comprefs peut être interpolée par une FLP. □

La somme des deux lemmes précédents constitue la preuve du Théorème 15.

Remarque 19. Le Théorème 15 peut être vu comme un renforcement du Théorème 1.8 et du Corollaire 6.2 de [4].

Notez que durant la preuve du Lemme 18 nous avons également démontré la proposition suivante.

Proposition 20. Soit $D = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\} \subseteq L^n$ et $f : D \to L$. Si pour tout $i, j \in [m]$ il existe $p^{i,j} \in \mathcal{P}_L^{(n)}$ qui coïncide avec f sur $\{\mathbf{x}^i, \mathbf{x}^j\}$, alors les FLP

$$\bigvee_{i=1}^{m} \bigwedge_{i=1}^{m} p^{i,j} \quad \text{et} \quad \bigwedge_{i=1}^{m} \bigvee_{i=1}^{m} p^{i,j} \tag{2.5}$$

interpolent f.

2.6 Interpolation des fonctions partielles

Dans toute cette section, on considère un treillis fini L, $D \subseteq L^n$ et $f : D \to L$. Nous décrivons une méthode pour calculer une FLP interpolant f, si une telle FLP existe. Pour cela nous définissons plusieurs algorithmes.

2.6.1 Représentations des FLP

Les FLP manipulées par nos algorithmes sont représentées en mémoire par le biais de structures de données (dictionnaires, listes, etc). Dans ce qui suit, on formalisera le lien entre les FLP et leurs représentations par une fonction ζ qui associe à chaque structure de données censée représenter une FLP la FLP correspondante. Pour toute structure de données `X` sur laquelle ζ est définie, on dira que `X` est une *représentation* de la FLP ζ(`X`).

Les représentations en mémoires des FLP et, de manière générale, les structures de données, seront dénotées par une police spéciale. Typiquement, les listes seront dénotées par `l` et les dictionnaires par des lettres majuscules (`P`, `W`, etc).

2.6.2 Calcul d'une FLP coïncidant avec f sur deux points

Dans la section précédente nous avons montré que f est une FLP partielle si et seulement si, pour chaque $\mathbf{x}, \mathbf{x'} \in D$, il existe une FLP coïncidant avec f sur $\{\mathbf{x}, \mathbf{x'}\}$. Voyons maintenant, pour des valeurs quelconques de $\mathbf{x}$ et $\mathbf{x'}$, comment calculer la représentation d'une telle FLP, s'il en existe une.

Dans toute cette sous-section, on considèrera deux élément $\mathbf{x}, \mathbf{x'} \in D$, et on utilisera l'expression « p produit (a, b) » pour signifier que la FLP p vérifie $(p(\mathbf{x}), p(\mathbf{x'})) = (a, b)$. Notez qu'une FLP produit $(f(\mathbf{x}), f(\mathbf{x'}))$ si et seulement si elle coïncide avec f sur $\{\mathbf{x}, \mathbf{x'}\}$.

Nous allons montrer comment calculer une FLP qui produit $(f(\mathbf{x}), f(\mathbf{x'}))$, si une telle FLP existe. Pour cela on se base sur le calcul de $\text{Я}_{\mathbf{x},\mathbf{x'}}$. Cet ensemble est égal à la fermeture de

$$\big\{(a,a) \;\big|\; a \in L\big\} \cup \big\{(x_i, x'_i) \;\big|\; i \in [n]\big\}$$

par les opérations $\wedge$ et $\vee$ définies sur L^2 (voir Remarque 8). Cette fermeture s'effectue par la procédure suivante :

1. $\text{Я} \leftarrow \big\{(a,a) \;\big|\; a \in L\big\} \cup \big\{(x_i, x'_i) \;\big|\; i \in [n]\big\}$,
2. tant qu'il existe $(a,b),(c,d) \in \text{Я}$ tels que

$$(a \wedge c, b \wedge d) \notin \text{Я} \qquad \big(\text{resp.} \quad (a \vee c, b \vee d) \notin \text{Я}\big),$$

   ajouter $(a \wedge c, b \wedge d)$ (resp. $(a \vee c, b \vee d)$) à Я.

Cette procédure termine en un nombre fini d'opérations, puisque L est fini. À la fin de cette procédure on a $\text{Я} = \text{Я}_{\mathbf{x},\mathbf{x'}}$.

Si $(f(\mathbf{x}), f(\mathbf{x'}))$ appartient à l'ensemble $\text{Я}_{\mathbf{x},\mathbf{x'}}$ ainsi construit, il existe une FLP qui produit $(f(\mathbf{x}), f(\mathbf{x'}))$. Afin de déterminer cette FLP : pendant la fermeture de Я, on mémorise une FLP qui produit chaque élément ajouté à Я. Pour tout (a,b) présent dans Я au moment de son initialisation, on connaît une FLP qui produit (a,b) puisque :

- si $(a,b) = (x_i, x'_i)$ où $i \in [n]$, alors (a,b) est produit par une projection,
- et si $a = b$, alors (a,b) est produit par une fonction constante.

De plus, pour toutes paires (a,b) et (c,d) respectivement produites par des FLP p et q, les paires $(a \wedge c, b \wedge d)$ et $(a \vee c, b \vee d)$ sont produites par $p \wedge q$ et $p \vee q$, respectivement.

L'Algorithme 1 implémente la fermeture de l'ensemble Я, qui est représenté par une liste $\texttt{l}_{\text{Я}}$. Pendant l'exécution de l'algorithme, les éléments de $\texttt{l}_{\text{Я}}$ sont combinés par les opérations $\wedge$ et $\vee$ et, lorsque de nouveaux éléments sont produits, ils sont ajoutés en queue de liste. Un dictionnaire `P` associe à chaque (a,b) présent dans $\texttt{l}_{\text{Я}}$ la représentation d'une FLP qui produit (a,b). Ce dictionnaire est complété au fur et à mesure que de nouvelles paires sont ajoutées à $\texttt{l}_{\text{Я}}$. La forme des représentations contenues dans `P` sera précisée plus loin. On dénote par $\texttt{P}[a,b]$ la valeur associée par `P` à la clé (a,b). La fonction SOUS-FLP renvoie une FLP qui produit $(f(\mathbf{x}), f(\mathbf{x'}))$, si une telle FLP existe.

Algorithme 1: Construction d'une FLP qui coïncide avec f sur $\{\mathbf{x}, \mathbf{x'}\}$, si une telle FLP existe (retourne null sinon).

Entrées: Un treillis L, une fonction partielle $f : D \to L$, deux tuples $\mathbf{x}, \mathbf{x'} \in D$.
Sorties: La représentation d'une FLP.

```
1  fonction SOUS-FLP(L, x, x', f)
2      (l_Я, P) ← INITIALISATION(L, x, x')
3      i ← 1
4      tant que i < taille(l_Я) faire
5          (a, b) ← l_Я[i]
6          pour j de 1 à i − 1 faire
7              (c, d) ← l_Я[j]
8              MAJ^∧(l_Я, P, (a, b), (c, d))
9              MAJ^∨(l_Я, P, (a, b), (c, d))
10         i ← i + 1
11     retourner P[f(x), f(x')]
```

Algorithme 2: Initialisation de la liste $\mathtt{l_Я}$ et d'un dictionnaire $\mathtt{P}$ qui associe à chaque couple de $\mathtt{l_Я}$ la FLP qui le produit.

Entrées: Un treillis L, deux tuples $\mathbf{x}, \mathbf{x'} \in L^n$.
Sorties: Une liste $\mathtt{l_Я}$ de couples, un dictionnaire $\mathtt{P}$ qui associe une FLP à chacun des couples de $\mathtt{l_Я}$.

```
1  fonction INITIALISATION(L, x, x')
2      l_Я ← liste vide
3      ∀a, b ∈ L : P[a, b] ← null
4      pour i ∈ [n] faire
5          si P[x_i, x'_i] = null alors
6              P[x_i, x'_i] ← « var i »
7              ajouter (x_i, x'_i) en queue de l_Я
8      pour a ∈ L faire
9          si P[a, a] = null alors
10             P[a, a] ← a
11             ajouter (a, a) en queue de l_Я
12     retourner (l_Я, P)
```

Algorithme 3: Mise à jour de $\mathtt{l_Я}$ et $\mathtt{P}$.

Entrées: La liste $\mathtt{l_Я}$, le dictionnaire $\mathtt{P}$, et deux couples $(a, b), (c, d)$.
Sorties: -

```
1  fonction MAJ^∧(l_Я, P, (a, b), (c, d))
2      (α, β) ← (a ∧ c, b ∧ d)
3      si P[α, β] = null alors
4          P[α, β] ← (P[a, b], ∧, P[c, d])
5          ajouter (α, β) en queue de l_Я
```

Algorithme 4: Mise à jour de $\mathtt{l_Я}$ et $\mathtt{P}$.

Entrées: La liste $\mathtt{l_Я}$, le dictionnaire $\mathtt{P}$, et deux couples $(a, b), (c, d)$.
Sorties: -

```
1  fonction MAJ^∨(l_Я, P, (a, b), (c, d))
2      (α, β) ← (a ∨ c, b ∨ d)
3      si P[α, β] = null alors
4          P[α, β] ← (P[a, b], ∨, P[c, d])
5          ajouter (α, β) en queue de l_Я
```

Les fonctions $\mathrm{MAJ}^\wedge$ et $\mathrm{MAJ}^\vee$ mettent à jour la liste $\mathtt{l_Я}$ et le dictionnaire $\mathtt{P}$. Les deux couples (a, b) et (c, d) sont combinés par $\wedge$ ou $\vee$ en un couple (α, β). Si (α, β) n'est pas encore une clé de $\mathtt{P}$, il est ajouté dans la liste $\mathtt{l_Я}$ et devient, dans $\mathtt{P}$, la clé associée à la représentation d'une FLP qui le produit.

Dans $\mathtt{P}$, chaque clé (a, b) est associée à la représentation d'une FLP qui la produit. Pour tout $a, b \in L$, $\mathtt{P}[a, b]$ représente la FLP $\zeta(\mathtt{P}[a, b])$ définie par :

- Si $\mathtt{P}[a, b] =$ « var i » (où $i \in [n]$), alors $\zeta(\mathtt{P}[a, b])$ est la projection de L^n à L définie par
$$\zeta(\mathtt{P}[a, b])(\mathbf{x}) = x_i.$$
- Si $\mathtt{P}[a, b] = a$ (où $a \in L$), alors $\zeta(\mathtt{P}[a, b])$ est la fonction constante de L^n à L définie par
$$\zeta(\mathtt{P}[a, b])(\mathbf{x}) = a.$$

- Si $\mathtt{P}[a,b] = (\mathtt{P}[c,d], \wedge, \mathtt{P}[c',d'])$, alors

$$\zeta(\mathtt{P}[a,b]) = \zeta(\mathtt{P}[c,d]) \wedge \zeta(\mathtt{P}[c',d'])$$

  et si $\mathtt{P}[a,b] = (\mathtt{P}[c,d], \vee, \mathtt{P}[c',d'])$, alors

$$\zeta(\mathtt{P}[a,b]) = \zeta(\mathtt{P}[c,d]) \vee \zeta(\mathtt{P}[c',d']).$$

Remarque 21. Lorsqu'un triplet est stockée dans $\mathtt{P}$ via l'instruction

$$\mathtt{P}[\alpha,\beta] \leftarrow (\mathtt{P}[a,b], \wedge, \mathtt{P}[c,d]), \quad (\text{resp. } \mathtt{P}[\alpha,\beta] \leftarrow (\mathtt{P}[a,b], \vee, \mathtt{P}[c,d])),$$

les valeurs de $\mathtt{P}[a,b]$ et $\mathtt{P}[c,d]$ ne sont pas copiées : le triplet stocké dans $\mathtt{P}[\alpha,\beta]$ contient deux références vers $\mathtt{P}[a,b]$ et $\mathtt{P}[c,d]$. En conséquence, l'espace nécessaire pour stocker $\mathtt{P}$ est $O(|L|^2)$.

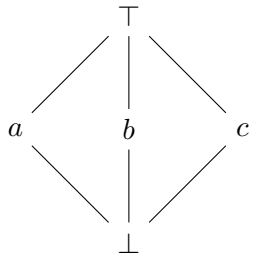


FIGURE 2.3 – Le treillis M_3.

Exemple 22. On considère le treillis M_3 décrit par la Figure 2.3, et la fonction $f : \{a,b\} \to M_3$ définie par

$$f(a) = b \quad \text{et} \quad f(b) = c.$$

La fonction INITIALISATION renvoie la liste suivante (l'ordre des éléments dans la liste est partiellement arbitraire)

$$\mathtt{l}_{\text{Я}} = \Big((a,b),(\bot,\bot),(\top,\top),(a,a),(b,b),(c,c)\Big).$$

Pendant l'exécution de la fonction SOUS-FLP, les éléments suivants sont ajoutés à la liste $\mathtt{l}_{\text{Я}}$ (dans l'ordre).

1 :	$(a,\bot)$	à partir de	$(a,b) \wedge (a,a)$	8 :	$(c,\bot)$	à partir de	$(c,c) \wedge (\top,b)$
2 :	$(a,\top)$	à partir de	$(a,b) \vee (a,a)$	9 :	$(b,\bot)$	à partir de	$(b,b) \wedge (\top,c)$
3 :	$(\bot,b)$	à partir de	$(a,b) \wedge (b,b)$	10 :	(a,c)	à partir de	$(a,\top) \wedge (\top,c)$
4 :	$(\top,b)$	à partir de	$(a,b) \vee (b,b)$	11 :	$(b,\top)$	à partir de	$(b,b) \vee (\bot,c)$
5 :	$(\top,c)$	à partir de	$(c,c) \vee (a,\bot)$	12 :	$(\bot,a)$	à partir de	$(a,a) \wedge (c,\top)$
6 :	$(\bot,c)$	à partir de	$(c,c) \wedge (a,\top)$	13 :	(c,b)	à partir de	$(\top,b) \wedge (c,\top)$
7 :	$(c,\top)$	à partir de	$(c,c) \vee (\bot,b)$	14 :	(b,c)	à partir de	$(b,\bot) \vee (\bot,c)$

La Figure 2.4 donne une description (partielle) de $\mathtt{P}$, où apparaissent tous les éléments de $\mathtt{P}$ qui composent la représentation de la FLP correspondant à $\mathtt{P}[b,c]$. Remarquez que la représentation obtenue de la FLP $\zeta(\mathtt{P}[b,c])$ est analogue aux représentations des fonctions booléennes sous formes de circuit [102] (les analogues des portes logiques utilisés ici étant $\wedge$ et $\vee$).

La FLP résultante peut être exprimée par

$$((x_1 \vee a) \wedge c) \vee ((c \vee (x_1 \wedge a)) \wedge b).$$

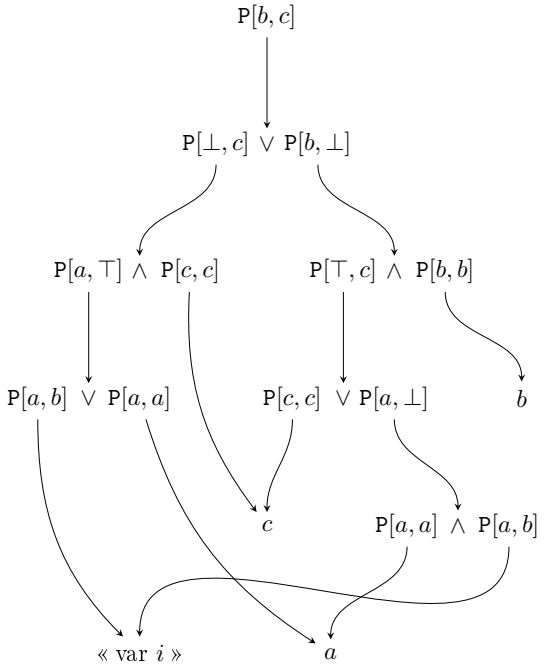


FIGURE 2.4 – Example 22 : représentation de la FLP qui produit (b, c) dans le dictionnaire P. Chaque flèche représente une association clé-valeur.

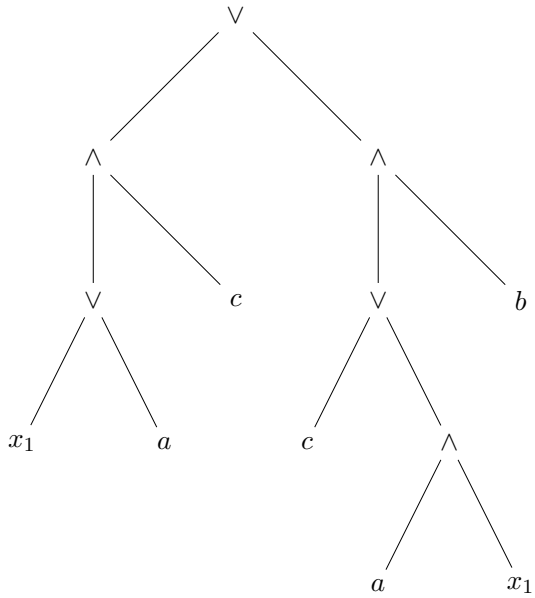


FIGURE 2.5 – Une représentation en arbre de la FLP correspondante

2.6.3 Calcul d'une FLP interpolant f

Soit $D = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\} \subseteq L^n$ et $f : D \to L$. Pour tout $i, j \in [m]$, soit $p^{i,j}$ une FLP qui coïncide avec f sur $\{\mathbf{x}^i, \mathbf{x}^j\}$. Soit $p^\vee$ et $p^\wedge$ les FLP définies par

$$p^\vee = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{m} p^{i,j} \quad \text{et} \quad p^\wedge = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} p^{i,j}.$$

On a montré dans la Section 2.5 que $p^\vee$ et $p^\wedge$ sont des interpolations de f. Puisqu'on sait calculer des représentations de $p^{i,j}$ pour tout $i, j \in [m]$, on peut aisément obtenir des représentations de $p^\vee$ et $p^\wedge$. Les fonctions INTERPOLATION$^\vee$ et INTERPOLATION$^\wedge$ retournent respectivement une représentation de $p^\vee$ et $p^\wedge$ sous forme de listes.

Algorithme 5: Calcul d'une représentation de $p^\vee$.

Entrées: Un treillis L, un ensemble $D = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\}$, une fonction $f : D \to L$.
Sorties: Une représentation d'une interpolation de f

```
1 fonction INTERPOLATION∨(L, D, f)
2     l∨ ← liste vide
3     pour i de 1 à m faire
4         l∧ ← liste vide
5         pour j de 1 à m faire
6             p ← SOUS-FLP(L, x^i, x^j, f)
7             ajouter p en queue de l∧
8         ajouter l∧ en queue de l∨
9     retourner l∨
```

Algorithme 6: Calcul d'une représentation de $p^\wedge$.

Entrées: Un treillis L, un ensemble $D = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\}$, une fonction $f : D \to L$.
Sorties: Une représentation d'une interpolation de f

```
1 fonction INTERPOLATION∧(L, D, f)
2     l∧ ← liste vide
3     pour i de 1 à m faire
4         l∨ ← liste vide
5         pour j de 1 à m faire
6             p ← SOUS-FLP(L, x^i, x^j, f)
7             ajouter p en queue de l∨
8         ajouter l∨ en queue de l∧
9     retourner l∧
```

Pour tout $\mathtt{l}_\vee$ retourné par INTERPOLATION$^\vee(L, D, f)$, la FLP $\zeta(\mathtt{l}_\vee)$ représentée par $\mathtt{l}_\vee$ est donnée par

$$\zeta(\mathtt{l}_\vee) = \bigvee_{\mathtt{l}_\wedge \in \mathtt{l}_\vee} \bigwedge_{\mathtt{p} \in \mathtt{l}_\wedge} \zeta(\mathtt{p}),$$

tandis que pour tout $\mathtt{l}_\wedge$ retourné par INTERPOLATION$^\wedge(L, D, f)$, la FLP $\zeta(\mathtt{l}_\wedge)$ représentée par $\mathtt{l}_\wedge$ est donnée par

$$\zeta(\mathtt{l}_\wedge) = \bigwedge_{\mathtt{l}_\vee \in \mathtt{l}_\wedge} \bigvee_{\mathtt{p} \in \mathtt{l}_\vee} \zeta(\mathtt{p}).$$

2.6.4 Complexité

Nous donnons maintenant des bornes supérieures des complexités de nos algorithmes (sous certaines conditions d'implémentation).

On suppose que, pour tout $\mathbf{x} \in D$, la valeur de $f(\mathbf{x})$ peut être obtenue en $O(1)$. On suppose également que L est implémenté par des tables (voir Section 1.6.1). Si cela n'est pas le cas, une implémentation de L sous forme de tables peut être créée en $O(|L|^2\Delta)$, où $O(\Delta)$ est la complexité des opérations $\wedge$ et $\vee$ dans l'implémentation originale du treillis L. Dans l'implémentation de

L sous forme de tables, les opérations $\wedge$ et $\vee$ s'effectuent en $O(1)$, et chaque élément de L est représenté par un entier.

Proposition 23. L'Algorithme 1 peut être implémenté de telle sorte que la fonction SOUS-FLP s'exécute en $O(|L|^4 + n)$.

Démonstration. Chaque chaque élément de L étant représenté par un entier, $\mathtt{P}$ peut être implémenté par un tableau de dimensions $|L| \times |L|$. L'accès et la modification des valeurs de $\mathtt{P}$ se fait alors en $O(1)$. On considère que $\mathtt{l}_{\mathtt{R}}$ est implémentée par une liste chaînée. On a alors les complexités suivantes.

Algorithme 2 : la fonction INITIALISATION s'exécute en $O(|L|^2 + n)$.

Algorithmes 3 et 4 : les fonctions MAJ$^\vee$ et MAJ$^\wedge$ s'exécute en $O(1)$.

Algorithme 1 : Chaque couple $(a, b) \in L$ ne peut être ajouté à $\mathtt{l}_{\mathtt{R}}$ qu'une seule fois. Par conséquent la taille de la liste est toujours au plus $|L|^2$. Le contenu de la boucle **pour** est donc exécuté moins de $|L|^4$ fois. Toutes les opérations contenues dans cette boucle s'exécutent en $O(1)$. En prenant en compte l'exécution de la fonction INITIALISATION avant la boucle principale, on obtient que la fonction SOUS-FLP s'exécute en

$$O(|L|^2 + n) + O(|L|^4) = O(|L|^4 + n).$$

□

Corollaire 24. Les Algorithmes 5 et 6 peuvent être implémentés de telle sorte que les fonctions INTERPOLATION$^\vee$ et INTERPOLATION$^\wedge$ s'exécutent en $O\big(|D|^2(|L|^4 + n)\big)$.

Le temps nécessaire à calculer $p(\mathbf{x})$ pour une FLP $p \in \mathcal{P}_L^{(n)}$ et $\mathbf{x} \in L^n$ dépend de la représentation de p. On donne donc des bornes supérieures de la complexité du calcul de $p(\mathbf{x})$ lorsque p est une FLP représentée par le résultat d'un de nos algorithme.

Proposition 25. Soit $p^{i,j}$ la FLP représentée par le résultat de SOUS-FLP$(L, \mathbf{x}^i, \mathbf{x}^j, f)$. Pour tout $\mathbf{x} \in L^n$, on peut calculer $p^{i,j}(\mathbf{x})$ en $O(|L|^2)$.

Démonstration. Soit $\mathtt{P}$ le dictionnaire construit durant l'exécution de SOUS-FLP$(L, \mathbf{x}^i, \mathbf{x}^j, f)$. La FLP $p^{i,j}$ est représentée par $\mathtt{P}[f(\mathbf{x}^i), f(\mathbf{x}^j)]$.

Soit $\mathbf{x} \in L^n$. Pour tout $a, b \in L$: Si $\zeta(\mathtt{P}[a, b])$ est une variable ou une constante, alors $\zeta(\mathtt{P}[a, b])(\mathbf{x})$ peut être obtenu en $O(1)$. Sinon, il existe deux clés (c, d) et (c', d') de $\mathtt{P}$ telles que $\zeta(\mathtt{P}[a, b])(\mathbf{x})$ est égale à l'infimum ou au supremum de $\zeta(\mathtt{P}[c, d])(\mathbf{x})$ et $\zeta(\mathtt{P}[c', d'])(\mathbf{x})$. Puisque le dictionnaire $\mathtt{P}$ contient au plus $|L|^2$ clés, un algorithme basé sur le principe de programmation dynamique peut calculer $\mathtt{P}[a, b](\mathbf{x})$ en $O(|L|^2)$. □

Corollaire 26. Soit $p^\vee$ et $p^\wedge$ les FLP représentées par les résultats respectifs de INTERPOLATION$^\vee(L, D, f)$ et INTERPOLATION$^\wedge(L, D, f)$. Pour tout $\mathbf{x} \in L^n$, on peut calculer $p^\vee(\mathbf{x})$ et $p^\wedge(\mathbf{x})$ en $O(|L|^2|D|^2)$.

2.7 Réduction de la taille des représentations

Dans toute cette section, on considère $D = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\} \subseteq L^n$, et une FLP partielle $f : D \to L$. Pour tout pour tout $i, j \in [m]$ on dénote par $\mathtt{p}^{i,j}$ le résultat de SOUS-FLP$(L, \mathbf{x}^i, \mathbf{x}^j, f)$, et par $p^{i,j}$ la FLP représentée par $\mathtt{p}^{i,j}$. Cette FLP coïncide avec f sur $\{\mathbf{x}^i, \mathbf{x}^j\}$.

Enfin, les résultats respectifs de INTERPOLATION$^\vee(L, D, f)$ et INTERPOLATION$^\wedge(L, D, f)$ sont dénotés par $\mathtt{p}^\vee$ et $\mathtt{p}^\wedge$, et représentent respectivement des FLP dénotées par $p^\vee$ et $p^\wedge$. Ces FLP vérifient

$$p^\vee = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{m} p^{i,j} \quad \text{et} \quad p^\wedge = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} p^{i,j}.$$

Notez que $\mathtt{p}^\vee$, tout comme $\mathtt{p}^\wedge$, est une liste de listes, et contient les représentations de m^2 « sous-FLP » ($\mathtt{p}^{i,j}$ pour chaque $i, j \in [m]$). Lorsque D est de taille conséquente, les expressions de $p^\vee$ et $p^\wedge$ ne sont donc pas lisibles par un humain.

Dans le but de compléter notre approche et de faciliter d'éventuelles applications, dans cette section nous abordons la réduction de la taille de $\mathtt{p}^\vee$ et $\mathtt{p}^\wedge$. Nous traitons cette problématique sous deux angles différents. Premièrement, nous montrons comment obtenir une représentation $\mathtt{p}^{i,j}$ de taille minimale en faisant appel à une variante de la fonction SOUS-FLP. Deuxièmement, nous verrons comment $\mathtt{p}^\vee$ et $\mathtt{p}^\wedge$ peuvent être simplifiées via la suppression de certaines sous-FLP. Cette réduction de la taille des représentations peut servir deux objectifs : permettre la lisibilité de la représentation résultante, et/ou réduire le temps nécessaire au calcul de $p^\vee(\mathbf{x})$ et $p^\wedge(\mathbf{x})$, pour $\mathbf{x} \in D$.

2.7.1 Minimisation de la taille de la représentation de $p^{i,j}$

Soit un dictionnaire $\mathtt{P}$ et $a, b \in L$ tels que $\mathtt{P}[a, b]$ représente une FLP. On définit la *taille* de $\mathtt{P}[a, b]$, dénotée par $\omega(\mathtt{P}[a, b])$, de la manière suivante

- si $\mathtt{P}[a, b]$ est une valeur de L, alors $\omega(\mathtt{P}[a, b]) = 1$,
- si $\mathtt{P}[a, b] =$ « var i » où $i \in [n]$ alors $\omega(\mathtt{P}[a, b]) = 1$,
- si $\mathtt{P}[a, b] = (\mathtt{P}[c, d], \wedge, \mathtt{P}[c', d'])$ ou si $\mathtt{P}[a, b] = (\mathtt{P}[c, d], \vee, \mathtt{P}[c', d'])$, alors

$$\omega(\mathtt{P}[a, b]) = \omega(\mathtt{P}[c, d]) + \omega(\mathtt{P}[c', d']) + 1.$$

Notez que cette définition de la taille de $\mathtt{P}[a, b]$ correspond au nombre de nœuds dans la représentation en arbre de la FLP représentée par $\mathtt{P}[a, b]$.

Il est possible de calculer la plus petite $\mathtt{p}^{i,j}$ représentant une FLP qui coïncide avec f sur $\{\mathbf{x}^i, \mathbf{x}^j\}$. Pour cela on fait appel à une variante des Algorithmes 1, 2, 3 et 4, où la taille de chaque valeur de $\mathtt{P}$ est mémorisée dans un second dictionnaire $\mathtt{W}$, et où la liste $\mathtt{l}_\mathcal{Я}$ est triée de telle sorte que $\omega(P[\mathtt{l}_\mathcal{Я}[i]]) \leq \omega(P[\mathtt{l}_\mathcal{Я}[j]])$ pour tout $i < j$. Pour cela, on remplace l'ajout en queue de liste utilisé dans les Algorithmes 3 et 4 par une insertion qui maintient la liste triée.

Cette seconde version de l'algorithme est disponible dans l'Annexe A.1. Notez que pour ce nouvel algorithme on a une complexité de $O(|L|^4 \log_2 |L| + n)$. Cette complexité supérieure à celle de SOUS-FLP est due au coût de l'insertion de nouveaux couples dans $\mathtt{l}_\mathcal{Я}$.

2.7.2 Réduction du nombre de sous-FLP

Nous abordons maintenant la simplification de $\mathtt{p}^\vee$ (la simplification $\mathtt{p}^\wedge$ peut être traitée de manière duale).

On cherche des ensembles $J_1, \ldots, J_m \subseteq [m]$ de tailles minimales, tels que la FLP $q^\vee$ définie par

$$q^\vee = \bigvee_{i=1}^{m} \bigwedge_{j \in J_i} p^{i,j}$$

est une interpolation de f. Afin d'aborder le problème du choix des ensembles $J_1, \ldots, J_m$ avec plus de facilité, on formule une contrainte sur les ensembles $J_1, \ldots, J_m$ garantissant que $\bigvee_{i=1}^{m} \bigwedge_{j \in J_i} p^{i,j}$ est une interpolation de f. Notez que cette condition est suffisante mais non nécessaire.

Proposition 27. La condition

$$\forall i, k \in [m] \; \exists j \in J_i \quad p^{i,j}(\mathbf{x}^k) \leq f(\mathbf{x}^k) \tag{2.6}$$

implique que $\bigvee_{i=1}^{m} \bigwedge_{j \in J_i} p^{i,j}$ est une interpolation de f.

Démonstration. En effet, cette condition implique que

$$\forall i, k \in [m], \quad \bigwedge_{j \in J_i} p^{i,j}(\mathbf{x}^k) \leq f(\mathbf{x}^k)$$

et donc que

$$\forall k \in [m], \quad \bigvee_{i=1}^{m} \bigwedge_{j \in J_i} p^{i,j}(\mathbf{x}^k) \leq f(\mathbf{x}^k).$$

De plus la condition

$$\forall k \in [m], \quad \bigvee_{i=1}^{m} \bigwedge_{j \in J_i} p^{i,j}(\mathbf{x}^k) \geq f(\mathbf{x}^k)$$

est nécessairement vérifiée puisque

$$\forall k \in [m], \quad \bigwedge_{j \in J_k} p^{k,j}(\mathbf{x}^k) = f(\mathbf{x}^k).$$

Donc pour tout $k \in [m]$ on a $\bigvee_{i=1}^{m} \bigwedge_{j \in J_i} p^{i,j}(\mathbf{x}^k) = f(\mathbf{x}^k)$. □

On cherche donc les plus petits ensembles $J_1, \ldots, J_m$ vérifiant (2.6). Remarquez que le choix de chaque ensemble J_i peut être effectué de manière indépendante. On pose donc le problème suivant.

Problème de simplification. *Pour chaque $i \in [m]$, retourner un ensemble $J_i \subseteq [m]$ de taille minimale, tel que*

$$\forall k \in [m] \; \exists j \in J_i \quad p^{i,j}(\mathbf{x}^k) \leq f(\mathbf{x}^k).$$

Ce problème peut être réduit à m problèmes de couverture par ensembles (set cover problem).

Problème de couverture par ensembles. *Soit un ensemble A et une famille $\boldsymbol{B}$ de sous-ensembles de A telle que $\bigcup \boldsymbol{B} = A$. Retourner un sous-ensemble de $\boldsymbol{B}$ de taille minimale et dont l'union des éléments est égale à A.*

Le problème de couverture par ensemble est NP-complet [67] ; certaines méthodes permettent cependant d'approcher un résultat optimal en temps polynomial [98].

On exprime le problème de simplification comme m problèmes de couverture par ensembles. Pour chaque $i \in [m]$, on décrit le problème de couverture par ensembles suivant. Soit $A = [m]$. Pour tout $j \in [m]$, soit B_j l'ensemble défini par

$$B_j = \left\{ k \in [m] \;\middle|\; p^{i,j}(\mathbf{x}^k) \leq f(\mathbf{x}^k) \right\}$$

et soit

$$\mathbf{B} = \{B_1, \ldots, B_m\}.$$

Soit J_i une solution du problème de couverture par ensembles pour A et $\mathbf{B}$, c'est à dire un sous-ensemble de $\mathbf{B}$ de taille minimale vérifiant $\bigcup J_i = A$. L'ensemble $\{j \in [m] | B_j \in J_i\}$ est un sous-ensemble de $[m]$ de taille minimale vérifiant (2.6). Les ensembles $J_1, \ldots, J_m$ constituent donc une solution du problème de simplification.

Afin d'achever la simplification de $\mathbf{p}^\vee$, on cherche un ensemble $I \subseteq [m]$ de taille minimale tel que

$$\bigvee_{i \in I} \bigwedge_{j \in J_i} p^{i,j}$$

est une interpolation de f. On traite la recherche de cet ensemble de manière similaire à la recherche des ensembles $J_1, \ldots, J_m$. La proposition suivante se démontre de manière similaire à la Proposition 27.

Proposition 28. La condition

$$\forall k \in [m]\ \exists i \in I \quad \bigwedge_{j \in J_i} p^{i,j}(\mathbf{x}^k) \geq f(\mathbf{x}^k) \tag{2.7}$$

implique que $\bigvee_{i \in I} \bigwedge_{j \in J_i} p^{i,j}$ est une interpolation de f.

On formule un deuxième problème de simplification, qui dépend des ensembles $J_1, \ldots, J_m$ obtenus à l'étape précédente.

Problème de simplification 2. *Retourner un ensemble $I \subseteq [m]$ de taille minimale, tel que*

$$\forall k \in [m]\ \exists i \in I \quad \bigwedge_{j \in J_i} p^{i,j}(\mathbf{x}^k) \geq f(\mathbf{x}^k).$$

Ce deuxième problème de simplification s'exprime comme un seul problème de couverture par ensembles. Soit $A = [m]$. Pour tout $i \in [m]$, soit B_i l'ensemble défini par

$$B_i = \left\{ k \in [m] \;\middle|\; \bigwedge_{j \in J_i} p^{i,j}(\mathbf{x}^k) \leq f(\mathbf{x}^k) \right\}$$

et soit

$$\mathbf{B} = \{B_1, \ldots, B_m\}.$$

Soit I une solution du problème de couverture par ensembles pour A et $\mathbf{B}$: $\{i \in [m] | B_i \in I\}$ est alors un sous-ensemble de taille minimale qui vérifie 2.7.

2.8 Conclusion et problèmes ouverts

Dans ce chapitre, nous avons proposé une nouvelle caractérisation des FLP partielles. Sur la base de cette caractérisation nous avons défini une méthode pour calculer une FLP interpolant une fonction partielle f, et ce en temps polynomial par rapport à la taille du domaine de f, de l'arité de f et de la taille du treillis sur lequel f est définie. La représentation de la FLP obtenue peut être de taille importante ; afin de faciliter une éventuelle application de notre méthode, nous avons également proposé des méthodes pour réduire la taille de son résultat.

Voici quelques questions et pistes ouvertes par les contributions de ce chapitre.

2.8.1 Calculer une description de $\mathcal{I}(f)$

La méthode que nous avons décrite dans ce chapitre ne permet pas d'obtenir une description de $\mathcal{I}(f)$. Une description de $\mathcal{I}(f)$ peut être donnée, par exemple, par ses bornes supérieures et inférieures, qui correspondent respectivement à la plus petite et à la plus grande FLP interpolant f. On sait que de telles bornes existent puisque $\bigwedge \mathcal{I}(f)$ et $\bigvee \mathcal{I}(f)$ sont des FLP qui appartiennent nécessairement à $\mathcal{I}(f)$. De plus, toute FLP p telle que $\bigwedge \mathcal{I}(f) \leq p \leq \bigvee \mathcal{I}(f)$ est forcément une interpolation de f. Dans [24], une description explicite des bornes de $\mathcal{I}(f)$ est donnée pour le cas où L est distributif. Cependant, nous ne connaissons pas de méthode pour calculer ces bornes lorsque L est un treillis non distributif.

Une piste de réflexion peut être donnée par la proposition suivante.

Proposition 29. Si, pour tout $i, j \in [m]$, $p^{i,j}$ est la plus petite (resp. la plus grande) FLP qui coïncide avec f sur $\{\mathbf{x}^i, \mathbf{x}^j\}$, alors $p^\vee = p^\wedge$, et $p^\vee$ est la plus petite (resp. la plus grande) FLP interpolant f. [2]

Démonstration. Soit $p \in \mathcal{I}(f)$. Pour tout $i, j \in [m]$, la fonction p coïncide avec f sur $\{\mathbf{x}^i, \mathbf{x}^j\}$ et donc, puisque $p^{i,j}$ est la plus petite FLP qui coïncide avec f sur $\{\mathbf{x}, \mathbf{x'}\}$, on a $p^{i,j} \leq p$. Donc

$$p^\vee = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{m} p^{i,j} \leq p \quad \text{et} \quad p^\wedge = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m} p^{i,j} \leq p.$$

En conclusion, les FLP exprimées par $p^\vee$ et $p^\wedge$ sont inférieures ou égales à tout $p \in \mathcal{I}(f)$. □

Cependant, nous ne savons pas calculer la plus petite (ou la plus grande) FLP coïncidant avec f sur $\{\mathbf{x}^i, \mathbf{x}^j\}$. De plus, il n'est pas certain que les bornes de $\mathcal{I}(f)$ constituent une description « satisfaisante » de cet ensemble, pour des raisons qui sont données dans la sous-section suivante.

2.8.2 Comparaison des FLP

Les solutions au problème d'interpolation se trouvent dans l'ensemble $\mathcal{P}_L^{(n)}$. Cet ensemble est un treillis (cela découle directement de la définition des FLP), que nous explorons en combinant, par les opérations $\wedge$ et $\vee$, des FLP déjà connues (les seules FLP connues a priori étant les projections et les fonctions constantes).

La principale difficulté rencontrée dans l'exploration de $\mathcal{P}_L^{(n)}$ est la suivante : la manipulation des FLP doit nécessairement s'effectuer par l'intermédiaire de leurs représentations (par exemple, sous forme d'arbre), et nous ne connaissons pas de forme de représentation des FLP qui permette de déterminer, pour deux FLP p et q, si $p \leq q$ (ou, de manière équivalente, si $p \vee q = q$) autrement qu'en calculant $p(\mathbf{x})$ et $q(\mathbf{x'})$ pour chaque $\mathbf{x} \in L^n$.

On peut voir ces difficultés comme une conséquence du fait que les FLP définies sur des treillis quelconques n'ont pas de forme normale connue. Dans le chapitre suivant, nous verrons que dans les cas où L est distributif, les FLP peuvent être exprimées sous formes normales (disjonctives ou conjonctives), et que l'on peut notamment calculer les bornes de $\mathcal{I}(f)$ en temps polynomial par rapport à n, $|D|$ et $|L|$.

2. Merci à Tamas Waldhauser pour avoir fait cette observation.

Chapitre 3

Fonctions latticielles polynomiales sur les treillis distributifs bornés

3.1 Introduction

Le chapitre précédent présentait les FLP et exposait de premiers résultats concernant le problème d'interpolation. La méthode de résolution du problème d'interpolation qui y était proposée avait pour limite de ne renvoyer qu'une seule solution (pas nécessairement optimale du point de vue de la concision de la représentation). De plus, cette méthode ne permettait pas d'obtenir les bornes de $\mathcal{I}(f)$.

Dans ce chapitre nous nous intéresserons aux FLP définies sur des treillis distributifs bornés. Pour rappel, un treillis, L est distributif si les identités

$$(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c) \quad \text{et} \quad (a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c) \tag{3.1}$$

sont respectées pour tout $a, b, c \in L$. Nous verrons que, dans ce cadre plus restreint, chaque FLP peut être exprimée par un forme normale, ce qui permet d'approcher différemment le problème d'interpolation. Nous montrerons notamment que si L est un treillis distributif borné pouvant être plongé dans un produit fini de chaînes (ce qui est nécessairement le cas lorsque L est fini), alors il est possible de calculer les bornes de $\mathcal{I}(f)$ en temps polynomial.

Dans la totalité de ce chapitre, à moins que l'inverse ne soit précisé, L désignera un treillis distributif borné.

3.2 Préliminaires

Soit un entier positif n et une fonction $f : L^n \to L$. Les trois conditions suivantes sont équivalentes [26, 51]

- f est une FLP,
- il existe $\mu : 2^{[n]} \to L$ tel que
$$p(\mathbf{x}) = \bigvee_{I \subseteq [n]} \mu(I) \wedge \bigwedge_{i \in I} x_i, \tag{3.2}$$
- il existe $\nu : 2^{[n]} \to L$ tel que
$$p(\mathbf{x}) = \bigwedge_{I \subseteq [n]} \nu(I) \vee \bigvee_{i \in I} x_i. \tag{3.3}$$

Les expression (3.2) et (3.3) sont respectivement appelées *forme normale disjonctive* (*disjonctive normal form*, DNF) et *forme normale conjonctive* (*conjonctive normal form*, CNF) de p. Notez qu'une FLP peut avoir plusieurs DNF et plusieurs CNF.

Exemple 30. Afin d'illustrer comment chaque FLP peut être exprimée en forme normale, on considère la FLP $p : L^3 \to L$ définie par $p(\mathbf{x}) = x_1 \vee (x_2 \wedge (a \vee x_3))$, où $a \in L$. En utilisant les identités (3.1), on exprime p en DNF :

$$\begin{aligned} p(\mathbf{x}) &= x_1 \vee (x_2 \wedge (a \vee x_3)) \\ &= x_1 \vee (a \wedge x_2) \vee (x_2 \wedge x_3) \\ &= \bigvee_{I \subseteq [3]} \mu(I) \wedge \bigwedge_{i \in I} x_i \end{aligned}$$

où $\mu : 2^{[3]} \to L$ est la fonction définie par

$$\mu(I) = \begin{cases} \top \text{ si } I = \{1\} \text{ ou } I = \{2,3\}, \\ a \text{ si } I = \{2\}, \\ \bot \text{ sinon.} \end{cases}$$

On peut également exprimer p en CNF :

$$\begin{aligned} p(\mathbf{x}) &= x_1 \vee (x_2 \wedge (a \vee x_3)) \\ &= (x_1 \vee x_2) \wedge (a \vee x_1 \vee x_3) \\ &= \bigwedge_{I \subseteq [3]} \nu(I) \vee \bigvee_{i \in I} x_i \end{aligned}$$

où $\nu : 2^{[3]} \to L$ est la fonction définie par

$$\nu(I) = \begin{cases} \bot \text{ si } I = \{1,2\}, \\ a \text{ si } I = \{1,3\}, \\ \top \text{ sinon.} \end{cases}$$

Remarque 31. Notez que l'existence d'une DNF ou d'une CNF n'est pas garantie pour les FLP définies sur des treillis non distributif. Par exemple, considérons le treillis M_3 décrit par la Figure 3.1 et la FLP $p : M_3 \to M_3$ défini par

$$p(x) = ((x \wedge a) \vee c) \wedge ((c \wedge (x \vee a)) \vee b).$$

Cette FLP ne peut être exprimée ni en DNF ni en CNF. En effet, on a

$$p(a) = b \quad \text{et} \quad p(b) = c,$$

et on peut aisément vérifier qu'aucune FLP exprimée par une DNF ne satisfait ces égalités.

Intéressons-nous maintenant à l'ensemble des formes normales qui expriment une FLP p de L^n à L. On peut montrer [26, 76] que l'ensemble des fonctions $\mu : 2^{[n]} \to L$ qui spécifient une DNF de p appartiennent à un intervalle $[\mu_p^*, \mu_p]$, dont les bornes sont définies de la manière suivante. La borne supérieure μ_p de l'intervalle est la fonction $\mu_p : 2^{[n]} \to L$ définie par

$$\mu_p(I) = p(\mathbf{1}_I),$$

où $\mathbf{1}_I$ est le tuple de taille n dont la i-ème composante vaut $\top$ si $i \in I$, et vaut $\bot$ sinon. Pour toute fonction $\mu \in [\mu_p^*, \mu_p]$ qui spécifie une DNF de p, on a

$$\mu_p(I) = \bigvee_{J \subseteq I} \mu(J).$$

La fonction μ_p est la seule fonction de l'intervalle $[\mu_p^*, \mu_p]$ qui soit non décroissante, c'est à dire qui vérifie

$$I \subseteq J \quad \Longrightarrow \quad \mu_p(I) \leq \mu_p(J)$$

pour tout $I, J \subseteq [n]$. Par la suite, les fonctions d'ensemble non décroissantes seront simplement appelées FEND. La borne inférieure de l'intervalle $[\mu_p^*, \mu_p]$ est définie par

$$\mu_p^*(I) = \begin{cases} \mu_p(I) \text{ si } \mu_p(I) > \bigvee_{J \subset I} \mu_p(J), \\ \bot \text{ sinon.} \end{cases}$$

La fonction μ^* est parfois appelée *transformée de Möbius ordinale* de μ_p [74, 78].

De manière duale, les CNF d'une FLP peuvent être spécifiées par une fonction d'ensemble $\nu : 2^{[n]} \to L$ appartenant à l'intervalle $[\nu_p, \nu_p^*]$, dont la borne inférieure est une fonction d'ensemble non croissante définie par

$$\nu_p(I) = p(\mathbf{0}_I),$$

où $\mathbf{0}_I$ est le tuple de taille n dont la i-ème composante vaut $\bot$ si $i \in I$, et vaut $\top$ sinon. La fonction ν_p est la seule fonction de l'intervalle $[\nu_p, \nu_p^*]$ qui soit non croissante, c'est à dire qui vérifie :

$$I \subseteq J \quad \Longrightarrow \quad \nu_p(J) \leq \nu_p(I).$$

pour tout $I, J \subseteq [n]$. La borne supérieure ν_p^* est définie par

$$\nu_p^*(I) = \begin{cases} \nu_p(I) \text{ si } \nu_p(I) < \bigwedge_{J \subset I} \nu_p(J), \\ \top \text{ sinon.} \end{cases}$$

Pour toute fonction $\nu \in [\nu_p, \nu_p^*]$ qui spécifie une CNF de p on a

$$\nu_p(I) = \bigwedge_{J \subseteq I} \nu(J).$$

Remarque 32. Pour tout $I \subseteq [n]$ on a

$$\nu_p(I) = p(\mathbf{0}_I) = p(\mathbf{1}_{[n] \setminus I}) = \mu_p([n] \setminus I).$$

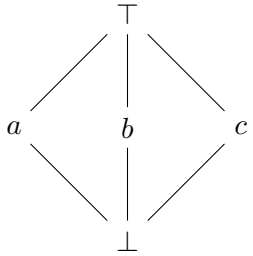


FIGURE 3.1 – Le treillis M_3.

En résumé, toute FLP définie sur un treillis distributif borné peut être spécifiée par une fonction d'ensemble [76]. Certaines de ces fonctions d'ensembles (μ^*, μ, ν^*, ν) définissent des expressions canonique de p. Définir les FLP par ces fonctions d'ensembles permet de s'abstraire de l'expression syntaxique de ces fonctions.

Enfin, on appelle *ensemble focal* de μ_p (resp. de ν_p) tout ensemble $I \subseteq [n]$ tel que

$$\mu_p(I) > \bigvee_{J \subset I} \mu_p(J) \qquad \Big(\text{resp.} \quad \nu_p(I) < \bigwedge_{J \subset I} \nu_p(J)\Big),$$

ou, de manière équivalente, tel que

$$\mu_p^*(I) \neq \bot \qquad \Big(\text{resp.} \quad \nu_p^*(I) \neq \top\Big).$$

On dénote respectivement par $\mathcal{F}(\mu_p)$ et $\mathcal{F}(\nu_p)$ les ensembles des ensembles focaux de μ_p des ν_p. Notez que p peut s'exprimer par une DNF ou CNF réduite, où les ensembles non focaux sont ignorés. Pour tout $\mathbf{x} \in L^n$ on a :

$$p(\mathbf{x}) = \bigvee_{I \in \mathcal{F}(\mu_p)} \mu_p(I) \wedge \bigwedge_{i \in I} x_i \quad \text{et} \quad p(\mathbf{x}) = \bigwedge_{I \in \mathcal{F}(\nu_p)} \nu_p(I) \vee \bigvee_{i \in I} x_i.$$

3.3 Représentation en mémoire des DNF et CNF

Soit une FLP p. Une DNF (respectivement, CNF) de p est donnée par n'importe laquelle des fonctions de l'intervalle $[\mu_p^*, \mu_p]$ (respectivement $[\nu_p, \nu_p^*]$). Une manière de représenter p en mémoire est donc de mémoriser l'une des fonctions de $[\mu_p^*, \mu_p]$ ou $[\nu_p, \nu_p^*]$.

Soit $\mu \in [\mu_p^*, \mu_p]$ et $\nu \in [\nu_p, \nu_p^*]$, et soit `M` et `N` deux dictionnaires représentant respectivement les fonctions et μ et ν en mémoire. Ces deux dictionnaires prennent pour clés des sous-ensembles de $[n]$, à chacun desquels ils associent une valeur de L. Pour tout $I \subseteq [n]$, si $\mu(I) > \bot$, alors $\texttt{M}[I] = \mu(I)$, sinon `M` ne contient pas la clé I. Pour tout $I \subseteq [n]$, si $\nu(I) < \top$ alors $\texttt{N}[I] = \nu(I)$, sinon `N` ne contient pas la clé I. À partir de ces dictionnaires, on peut calculer la valeur de $p(\mathbf{x})$ pour tout $\mathbf{x} \in L^n$ donné.

Algorithme 7: Calcule $p(\mathbf{x})$ à partir de $\mathbf{x}$ et `M`.

```
1 fonction OUTPUT_DNF(M, x)
2     y ← ⊥
3     pour chaque clé I de M faire
4         a ← M[I]
5         pour chaque i ∈ I faire
6             a ← a ∧ x_i
7         y ← y ∨ a
8     retourner y
```

Algorithme 8: Calcule $p(\mathbf{x})$ à partir de $\mathbf{x}$ et `N`.

```
1 fonction OUTPUT_CNF(N, x)
2     y ← ⊤
3     pour chaque clé I de M faire
4         a ← N[I]
5         pour chaque i ∈ I faire
6             a ← a ∨ x_i
7         y ← y ∧ a
8     retourner y
```

Au pire cas, ces deux algorithmes s'exécutent en $O(2^n n)$. Cependant, dans certains contextes, on peut garantir que le nombre de clés dans `M` et `N` est plus faible que 2^n, ce qui permet de donner une borne supérieure plus basse aux temps d'exécution de ces algorithmes. Notez que μ_p^* et ν_p^* sont les fonctions des intervalles $[\mu_p^*, \mu_p]$ et $[\nu_p, \nu_p^*]$ dont la représentation en mémoire nécessite le moins d'espace.

3.4 L'intégrale de Sugeno

L'intégrale de Sugeno est une intégrale introduite pour la première fois dans [91]. Elle appartient, avec l'intégrale de Choquet, à la classe des intégrales dites floues. Elle a depuis été étudiée, dans sa version discrète, en tant que *fonction d'agrégation* (voir [75]), et utilisée dans le domaine de la MCDA, pour sa capacité à fusionner des valeurs appartenant à un ensemble ordonné (le plus souvent, totalement ordonnés), et ainsi modéliser des relations de préférence sur des ensembles d'alternatives, lorsque les alternatives considérées sont évaluées sur plusieurs critères à travers des échelles non numériques (voir Introduction et [25, 42, 54]).

Dans cette section, nous rappelons la définition des intégrales de Sugeno (dans sa version discrète) et en quoi ces fonctions constituent une sous-classe de FLP.

Une intégrale de Sugeno $S_\mu : L^n \to L$ est une fonction d'agrégation définie par rapport à une capacité $\mu : 2^{[n]} \to L$. Une capacité μ est une FEND qui vérifie les conditions $\mu(\emptyset) = \bot$ et $\mu([n]) = \top$. L'intégrale de Sugeno est à l'origine définie sur des ensembles totalement ordonnés. Pour une chaîne L et une capacité $\mu : 2^{[n]} \to L$, on définit l'intégrale de Sugeno $S_\mu : L^{[n]} \to L$ par

$$S_\mu(\mathbf{x}) = \bigvee_{i=1}^{n} \Big(x_{\sigma(i)} \wedge \mu\big(\{\sigma(i), \ldots, \sigma(n)\}\big) \Big),$$

où $\sigma : [n] \to [n]$ est une bijection telle que $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \cdots \leq x_{\sigma(n)}$. De manière équivalente, on peut définir S_μ par

$$S_\mu(\mathbf{x}) = \bigvee_{I \subseteq [n]} \mu(I) \wedge \bigwedge_{i \in I} x_i.$$

Toute intégrale de Sugeno est donc une FLP. On peut donc généraliser de manière naturelle la définition des intégrales de Sugeno au cas où L est un treillis distributif borné [26].

Soit un treillis distributif borné L et une FLP $p : L^n \to L$. Les trois propositions suivantes sont équivalentes.

- p est une intégrale de Sugeno.
- μ_p vérifie $\mu_p(\emptyset) = \bot$ et $\mu_p([n]) = \top$ (autrement dit, μ_p est une capacité).
- p est idempotente, c'est à dire que $p(a, \ldots, a) = a$ pour tout $a \in L$.

3.5 Fonctions partielles et interpolation : état de l'art

Pour rappel, dans le chapitre précédent, on avait défini $\mathcal{I}(f)$ comme étant l'ensemble des FLP interpolant la fonction partielle $f : D \to L$, où $D \subseteq L^n$. Cet ensemble possède toujours une borne inférieure et une borne supérieure. Nous traitons maintenant le cas où L est un treillis distributif borné, et nous nous intéressons aux questions suivantes :

- Comment caractériser les bornes de $\mathcal{I}(f)$?
- Quelles procédures algorithmiques mettre en œuvre pour calculer ces bornes ? Quelle est la complexité de ces procédures ?

Dans cette section, nous rappelons les réponses apportées à ces questions par des travaux antérieurs.

3.5.1 Cas général

Dans [24], les bornes de $\mathcal{I}(f)$ sont caractérisées de la manière suivante. D'abord, le treillis L est plongé dans un treillis booléen B, et f est traitée comme une fonction partielle de B^n vers B. Une formulation explicite des bornes de l'ensemble des solutions au problème d'interpolation sur B (on appellera cet ensemble $\mathcal{I}_B(f)$) est donnée. Enfin, les bornes de $\mathcal{I}(f)$ sont exprimées en fonctions des bornes de $\mathcal{I}_B(f)$.

Plongement de L. Puisque L est distributif et borné, il est possible de l'envisager comme un sous-treillis d'un treillis Booléen (voir Sous-section 1.5.2 pour le cas où L est fini ; pour le cas où L est infini, voir le théorème de représentation de Birkhoff-Priestley [36]). Soit B un treillis Booléen tel que L est un sous-treillis de B et tel que $\bot_L = \bot_B$ et $\top_L = \top_B$. Puisque B est Booléen, chaque $x \in B$ possède un complément $\overline{x} \in B$, qui est l'unique élément tel que $x \wedge \overline{x} = \bot_B$ et $x \vee \overline{x} = \top_B$. Notez que tout élément de L possède un complément dans B, qui n'est pas nécessairement dans L.

FIGURE 3.2 – À gauche : un treillis L. À droite, un treillis Booléen dont L est un sous-treillis.

Caractérisation des bornes de $\mathcal{I}_B(f)$. Afin d'obtenir les bornes de $\mathcal{I}_B(f)$, les FEND $\mu^- : 2^{[n]} \to B$ et $\mu^+ : 2^{[n]} \to B$ sont définies par

$$\mu^-(I) = \bigvee_{\mathbf{x} \in D} \Big(f(\mathbf{x}) \wedge \bigwedge_{i \notin I} \overline{x_i} \Big) \quad \text{et} \quad \mu^+(I) = \bigwedge_{\mathbf{x} \in D} \Big(f(\mathbf{x}) \vee \bigvee_{i \in I} \overline{x_i} \Big).$$

L'ensemble $\mathcal{I}_B(f)$ est non vide si et seulement si $\mu^- \leq \mu^+$. Dans ce cas, les FLP $p^- : B^n \to B$ et $p^+ : B^n \to B$ définies par

$$p^- = \bigvee_{I \subseteq [n]} \mu^-(I) \wedge \bigwedge_{i \in I} x_i \quad \text{et} \quad p^+ = \bigvee_{I \subseteq [n]} \mu^+(I) \wedge \bigwedge_{i \in I} x_i$$

sont les bornes de $\mathcal{I}_B(f)$.

Caractérisation des bornes de $\mathcal{I}(f)$. On regarde maintenant $\mathcal{I}(f)$ comme l'ensemble qui contient toutes les FLP $p \in \mathcal{I}_B(f)$ qui vérifient

$$\forall \mathbf{x} \in L^n, \quad p(\mathbf{x}) \in L,$$

ou, de manière équivalente

$$\forall I \subseteq [n], \quad \mu_p(I) \in L. \tag{3.4}$$

On cherche donc la plus petite et la plus grande FEND vérifiant (3.4) et comprise entre μ^- et μ^+. Pour cela on fait appel aux fonctions $\mathrm{cl} : B \to L$ et $\mathrm{int} : B \to L$, qui sont définies par

$$\mathrm{cl}(x) = \bigwedge\{y \in L \mid x \leq y\} \quad \text{et} \quad \mathrm{int}(x) = \bigvee\{y \in L \mid y \leq x\}.$$

On dénote par μ^{cl} et μ^{int} les FEND de $2^{[n]}$ à L définies par

$$\mu^{\mathrm{cl}}(I) = \mathrm{cl}\big(\mu^-(I)\big) \quad \text{et} \quad \mu^{\mathrm{int}}(I) = \mathrm{int}\big(\mu^+(I)\big).$$

Les fonctions μ^{cl} et μ^{int} sont, respectivement, la plus petite et la plus grande FEND comprise entre μ^- et μ^+ vérifiant (3.4). L'ensemble $\mathcal{I}(f)$ est non vide si et seulement si $\mu^{\mathrm{cl}} \leq \mu^{\mathrm{int}}$. Dans ce cas, les FLP $p^{\mathrm{cl}} : L^n \to L$ et $p^{\mathrm{int}} : L^n \to L$ définies par

$$p^{\mathrm{cl}}(\mathbf{x}) = \bigvee_{I \subseteq [n]} \mu^{\mathrm{cl}}(I) \wedge \bigwedge_{i \in I} x_i \quad \text{et} \quad p^{\mathrm{int}}(\mathbf{x}) = \bigvee_{I \subseteq [n]} \mu^{\mathrm{int}}(I) \wedge \bigwedge_{i \in I} x_i$$

sont les bornes de $\mathcal{I}(f)$.

Remarque 33. Il est possible que f puisse être interpolée par une FLP sur B et non sur L.

Remarque 34. Notez que la description des bornes μ^{cl} et μ^{int} de $\mathcal{I}(f)$ donnée plus haut repose sur un calcul explicite des valeurs $\mu^-(I)$ et $\mu^+(I)$ pour chaque $I \subseteq [n]$. Puisque l'ensemble $[n]$ possède 2^n sous-ensembles, le calcul de $\mu^-(I)$ pour chaque $I \subseteq [n]$ correspond à 2^n calculs dont la complexité est $O(|D|n\Delta)$, où $O(\Delta)$ est la complexité des opérations $\wedge$ et $\vee$.

De même, la meilleure méthode que nous connaissons pour tester si $\mu^{\mathrm{cl}} \leq \mu^{\mathrm{int}}$ s'exécute en $\Omega(2^n)$ au pire cas. L'application de la méthode de calcul des bornes de $\mathcal{I}(f)$ présentée dans [24] est donc limitée à l'interpolation de fonctions de faible arité.

3.5.2 Lorsque L est un ensemble totalement ordonné

Dans le cas où L est une chaîne, les bornes de $\mathcal{I}(f)$ peuvent être caractérisées plus simplement, et leur calcul peut être effectué en temps polynomial par rapport à $|D|$, n et L.

Remarque 35. Les résultats exposés ici sont reformulés d'après [83] et [86], et serviront de base aux démonstrations de la section suivante. Ces résultats concernent à l'origine l'intégrale de Sugeno discrète, mais peuvent aisément être généralisés aux FLP. Dans un but principalement didactique, nous fournissons une preuve pour chaque proposition de cette sous-section.

Soit une chaîne L et une fonction $f : D \to L$ (où $D \subseteq L^n$). Tout d'abord, voyons sous quelles conditions une FLP p vérifie $p(\mathbf{x}) = f(\mathbf{x})$, pour un $\mathbf{x} \in L^n$ donné. On définit les ensembles suivants.

$$I^{>}_{f,\mathbf{x}} = \big\{i \in [n] \;\big|\; x_i > f(\mathbf{x})\big\} \quad \text{et} \quad I^{\geq}_{f,\mathbf{x}} = \big\{i \in [n] \;\big|\; x_i \geq f(\mathbf{x})\big\}.$$

Proposition 36. L'égalité $p(\mathbf{x}) = f(\mathbf{x})$ est vérifiée si et seulement si

$$\mu_p\big(I^{>}_{f,\mathbf{x}}\big) \leq f(\mathbf{x}) \leq \mu_p\big(I^{\geq}_{f,\mathbf{x}}\big). \tag{3.5}$$

Démonstration. Voyons d'abord pourquoi la condition (3.5) est nécessaire. Supposons que (3.5) n'est pas vérifiée. Si $f(\mathbf{x}) < \mu_p(I^{>}_{f,\mathbf{x}})$, alors nécessairement $f(\mathbf{x}) < p(\mathbf{x})$. Si $\mu_p(I^{\geq}_{f,\mathbf{x}}) < f(\mathbf{x})$ alors pour tout $I \subseteq [n]$:

- si $I \subset I^{\geq}_{f,\mathbf{x}}$ alors $\mu_p(I) < f(\mathbf{x})$, car μ_p est non décroissante,

- sinon on a $I \not\subseteq I^{\geq}_{f,\mathbf{x}}$ et alors il existe $i \in I$ tel que $x_i < y$.

Pour tout $I \subseteq [n]$ on a donc

$$\mu_p(I) \wedge \bigwedge_{i \in I} x_i < f(\mathbf{x}).$$

Donc $\mu_p(I^{\geq}_{f,\mathbf{x}}) < f(\mathbf{x})$ implique $p(\mathbf{x}) < f(\mathbf{x})$. Voyons maintenant pourquoi la condition est suffisante. On suppose que (3.5) est vérifiée. Pour tout $I \subseteq [n]$:

- si $I \subseteq I^{>}_{f,\mathbf{x}}$ alors $\mu_p(I) \leq f(\mathbf{x})$,
- sinon on a $I \not\subseteq I^{>}_{f,\mathbf{x}}$ et alors il existe $i \in I$ tel que $x_i \leq f(\mathbf{x})$.

Donc pour tout $I \subseteq [n]$

$$\mu_p(I) \wedge \bigwedge_{i \in I} x_i \leq f(\mathbf{x}).$$

De plus, on a nécessairement

$$\mu\big(I^{\geq}_{f,\mathbf{x}}\big) \wedge \bigwedge_{i \in I^{\geq}_{f,\mathbf{x}}} x_i \geq f(\mathbf{x}).$$

On a donc bien $p(\mathbf{x}) = f(\mathbf{x})$. □

Corollaire 37. La fonction p est une interpolation de f si et seulement si μ_p vérifie

$$\mu_p\big(I^{>}_{f,\mathbf{x}}\big) \leq f(\mathbf{x}) \leq \mu_p\big(I^{\geq}_{f,\mathbf{x}}\big) \tag{3.6}$$

pour tout $\mathbf{x} \in D$.

Étant donnée $f : D \to L$ (où $D \subseteq L^n$), comment savoir s'il existe une FEND qui satisfait (3.6) pour chaque $\mathbf{x} \in D$? La proposition suivante donne plusieurs conditions nécessaires et suffisantes.

Proposition 38. Les quatre conditions suivantes sont équivalentes.

1. La fonction $f : D \to L$ est une FLP partielle.
2. Il existe une FEND μ qui vérifie,

$$\mu\big(I^{>}_{f,\mathbf{x}}\big) \leq f(\mathbf{x}) \leq \mu\big(I^{\geq}_{f,\mathbf{x}}\big) \quad \text{et} \quad \mu\big(I^{>}_{f,\mathbf{x'}}\big) \leq f(\mathbf{x'}) \leq \mu\big(I^{\geq}_{f,\mathbf{x'}}\big) \tag{3.7}$$

   pour tout $\mathbf{x}, \mathbf{x'} \in D$.
3. La fonction f vérifie

$$f(\mathbf{x}) < f(\mathbf{x'}) \quad \Longrightarrow \quad I^{\geq}_{f,\mathbf{x}}, \not\subseteq I^{>}_{f,\mathbf{x}}$$

   pour tout $\mathbf{x}, \mathbf{x'} \in D$.
4. La fonction f vérifie

$$f(\mathbf{x}) < f(\mathbf{x'}) \quad \Longrightarrow \quad \exists i \in [n],\ x_i \leq f(\mathbf{x}) < f(\mathbf{x'}) \leq x'_i$$

   pour tout $\mathbf{x}, \mathbf{x'} \in D$ [30].

Démonstration. On sait qu'il existe une FLP interpolant f si et seulement si, pour tout $\mathbf{x}, \mathbf{x'} \in D$, il existe une FLP qui coïncide avec f sur $\{\mathbf{x}, \mathbf{x'}\}$ (voir Théorème 15). Donc, en se basant sur la Proposition 36, on obtient que les conditions 1 et 2 sont équivalentes.

L'équivalence entre les conditions 3 et 4 découle directement des définitions de $I^{\geq}_{f,\mathbf{x}}$ et $I^{>}_{f,\mathbf{x}}$.

Afin de montrer que les conditions 2 et 3 sont équivalentes, supposons d'abord que la condition 3 n'est pas vérifiée, c'est à dire qu'il existe $\mathbf{x}, \mathbf{x'} \in D$ tels que : $f(\mathbf{x}) < f(\mathbf{x'})$ et $I^{\geq}_{f,\mathbf{x'}} \subseteq I^{>}_{f,\mathbf{x}}$. Pour toute fonction μ vérifiant (3.7) on a

$$\mu(I^{>}_{f,\mathbf{x}}) \leq f(\mathbf{x}) < f(\mathbf{x'}) \leq \mu(I^{\geq}_{f,\mathbf{x'}}),$$

et donc μ n'est pas non décroissante. Donc il n'existe aucune FEND qui vérifie la contrainte (3.7).

Soit μ la FEND définie par

$$\mu(I) = \bigvee \{f(\mathbf{x}) \mid \mathbf{x} \in D,\ I^{\geq}_{f,\mathbf{x}} \subseteq I\}.$$

Notez qu'on considère $\bigvee \emptyset = \bot$. On montre que μ vérifie (3.6) pour tout $\mathbf{x} \in D$. Soit $\mathbf{x} \in D$. Clairement on a $\mu(I^{\geq}_{f,\mathbf{x}}) \geq f(\mathbf{x})$. De plus, puisque la condition 3 est vérifiée, on a

$$\forall \mathbf{x'} \in D, \quad I^{\geq}_{f,\mathbf{x'}} \subseteq I^{>}_{f,\mathbf{x}} \quad \Longrightarrow \quad f(\mathbf{x'}) \leq f(\mathbf{x}),$$

et donc

$$\mu(I^{>}_{f,\mathbf{x}}) = \bigvee \{f(\mathbf{x'}) \mid \mathbf{x'} \in D,\ I^{\geq}_{f,\mathbf{x'}} \subseteq I^{>}_{f,\mathbf{x}}\} \leq f(\mathbf{x}).$$

□

3.6 Nouvelle caractérisation des bornes de $\mathcal{I}(f)$

3.6.1 Caractérisation des bornes

Les propositions de la section précédente permettent de calculer les bornes de $\mathcal{I}(f)$ et de vérifier si cet ensemble est vide en temps polynomial. Nous généralisons ces résultats au cas où L est un treillis distributif borné pouvant être plongé dans un produit fini de chaînes. Pour cela nous nous appuyons sur la notion de plongement de Dilworth (voir Sous-section 1.5.3), qui garantit l'existence d'un plongement de L dans un produit de chaîne.

On considère un plongement de Dilworth de L dans un produit de k chaînes, dénoté par $\gamma : L \to L_1 \times \cdots \times L_k$. Pour tout $x \in L$, on dénote par $x_{|_i}$ la i-ème composante de $\gamma(x)$. Autrement dit,

$$\gamma(x) = \left(x_{|_1}, \ldots, x_{|_k}\right) \in L_1 \times \cdots \times L_k.$$

Comme expliqué dans 1.5.3, pour tout $a, b \in L$ on a

$$\gamma(a \wedge b) = \gamma(a) \wedge \gamma(b) = \left(a_{|_1} \wedge b_{|_1}\ ,\ \ldots\ ,\ a_{|_k} \wedge b_{|_k}\right)$$
$$\gamma(a \vee b) = \gamma(a) \vee \gamma(b) = \left(a_{|_1} \vee b_{|_1}\ ,\ \ldots\ ,\ a_{|_k} \vee b_{|_k}\right)$$

ainsi que

$$a \leq b \quad \Longleftrightarrow \quad \forall i \in [k],\ a_{|_i} \leq b_{|_i} \quad .$$

Soit $D \subseteq L^n$ et une fonction $f : D \to L$. On commence par définir les valeurs suivantes. Pour chaque $y \in L$ et $t \in [k]$,

$$s^{-}_{y,t} = \bigwedge \left\{a \in L \;\middle|\; a_{|_t} = y_{|_t}\right\} \quad \text{et} \quad s^{+}_{y,t} = \bigvee \left\{a \in L \;\middle|\; a_{|_t} = y_{|_t}\right\}.$$

Autrement dit, $s^{-}_{y,t}$ et $s^{+}_{y,t}$ sont respectivement le plus petit et le plus grand élément de L qui est égal à y sur la t-ième composante. Pour chaque $\mathbf{x} \in D$ et $t \in [k]$,

$$I^{\geq}_{f,\mathbf{x},t} = \left\{i \;\middle|\; x_{i|_t} \geq f(\mathbf{x})_{|_t}\right\} \quad \text{et} \quad I^{>}_{f,\mathbf{x},t} = \left\{i \;\middle|\; x_{i|_t} > f(\mathbf{x})_{|_t}\right\}.$$

Proposition 39. Soit une FLP $p : L^n \to L$. Pour tout $\mathbf{x} \in L^n$, on a $p(\mathbf{x}) = f(\mathbf{x})$ si et seulement si les inégalités

$$s^-_{f(\mathbf{x}),t} \leq \mu_p\Big(I^{\geq}_{f,\mathbf{x},t}\Big) \quad \text{et} \quad \mu_p\Big(I^{>}_{f,\mathbf{x},t}\Big) \leq s^+_{f(\mathbf{x}),t}$$

sont vérifiées pour tout $t \in [k]$.

Démonstration. Tout d'abord, remarquez que

$$\bigvee_{I \subseteq [n]} \mu_p(I) \wedge \bigwedge_{i \in I} x_i = f(\mathbf{x})$$

est vérfiée si et seulement si, pour tout $t \in [k]$, on a

$$\bigvee_{I \subseteq [n]} \mu_p(I)_{|_t} \wedge \bigwedge_{i \in I} x_{i|_t} = f(\mathbf{x})_{|_t} \quad . \tag{3.8}$$

La partie gauche de (3.8) exprime une FLP de $(L_t)^n$ à L_t, qu'on appelle p_t telle que

$$\mu_{p_t}(I) = \mu_p(I)_{|_t}$$

pour tout $I \subseteq [n]$. Puisque l'égalité (3.8) est équivalente à

$$p_t\big(x_{1|_t}, \ldots, x_{n|_t}\big) = f(\mathbf{x})_{|_t}$$

et que la fonction p_t est une FLP sur la chaine L_t, en appliquant la Proposition 36 on obtient que (3.8) est équivalent à

$$f(\mathbf{x})_{|_t} \leq \mu_p\big(I^{\geq}_{f,\mathbf{x},t}\big)_{|_t} \quad , \tag{3.9}$$

$$\mu_p\big(I^{>}_{f,\mathbf{x},t}\big)_{|_t} \leq f(\mathbf{x})_{|_t} \quad . \tag{3.10}$$

Puisque $s^-_{f(\mathbf{x}),t}$ est le plus petit élément de L dont la t-ième composante est égale à $f(\mathbf{x})_{|_t}$, (3.9) équivaut à $s^-_{f(\mathbf{x}),t} \leq \mu(I^{\geq}_{f,\mathbf{x},t})$. De manière duale, puisque $s^+_{f(\mathbf{x}),t}$ est le plus grand élément de L dont la t-ième composante est égale à $f(\mathbf{x})_{|_t}$, (3.10) équivaut à $\mu(I^{>}_{f,\mathbf{x},t}) \leq s^+_{f(\mathbf{x}),t}$. □

Remarque 40. Cette proposition généralise la Proposition 36 aux cas où L est un treillis distributif borné pouvant être plongé dans un produit fini de chaînes. En effet, lorsque L est une chaine, et que le plongement de Dilworth considéré est la fonction identité, la Proposition 39 devient équivalente à la Proposition 36.

Corollaire 41. Une FLP $p : L^n \to L$ appartient à $\mathcal{I}(f)$ si et seulement si, pour tout $\mathbf{x} \in D$ et tout $t \in [k]$,

$$s^-_{f(\mathbf{x}),t} \leq \mu_p\Big(I^{\geq}_{f,\mathbf{x},t}\Big) \quad \text{et} \quad \mu_p\Big(I^{>}_{f,\mathbf{x},t}\Big) \leq s^+_{f(\mathbf{x}),t} \quad .$$

Les contraintes données par le corollaire ci-dessus caractérisent l'ensemble des FLP interpolant f. On appelle *contraintes minimales* les contraintes de la forme $s^- \leq \mu(I)$ (où $s^- \in L$) et *contraintes maximales* les contraintes de la forme $\mu(I) \leq s^+$ (où $s^+ \in L$). On représentent les contraintes minimales et les contraintes maximales qui caractérisent f par deux ensembles, respectivement

$$C^-_f = \Big\{\Big(I^{\geq}_{f,\mathbf{x},t}\ ,\ s^-_{f(\mathbf{x}),t}\Big) \ \Big|\ \mathbf{x} \in D,\ t \in [k]\Big\} \quad \text{et} \quad C^+_f = \Big\{\Big(I^{>}_{f,\mathbf{x},t}\ ,\ s^+_{f(\mathbf{x}),t}\Big) \ \Big|\ \mathbf{x} \in D,\ t \in [k]\Big\}.$$

Les contraintes caractérisant toutes les FLP $p \in \mathcal{I}(f)$ sont exprimées par rapport à la fonction μ_p. Cependant, puisque $\mu_p(I) = \nu_p([n]\backslash I)$ (voir Remarque 32), toute condition de la forme

$$s^- \leq \mu_p\big(I^{\geq}\big) \qquad \Big(\text{resp.} \quad \mu_p\big(I^{>}\big) \leq s^+\Big)$$

est équivalente à la condition exprimée par rapport à ν_p sous la forme

$$s^- \leq \nu_p\big([n]\backslash I^{\geq}\big) \qquad \Big(\text{resp.} \quad \nu_p\big([n]\backslash I^{>}\big) \leq s^+\Big).$$

3.6.2 Identification des FLP partielles

Il se peut que C_f^- et C_f^+ décrivent un ensemble de contraintes qu'aucune FEND ne satisfait. Dans ce cas (et seulement dans ce cas), $\mathcal{I}(f)$ est vide. Cette situation peut être identifiée en testant indépendamment la compatibilité de chaque contrainte minimale (décrite par un élément de C_f^-) avec chaque contrainte maximale (décrite par un élément de C_f^+).

Proposition 42. Les assertions suivantes sont équivalentes.

1. La fonction f est une FLP partielle.
2. Il existe une FEND $\mu : 2^{[n]} \to L$ telle que

$$\forall\big(I^{\geq}, s^-\big) \in C_f^- \quad s^- \leq \mu\big(I^{\geq}\big) \qquad \text{et} \qquad \forall\big(I^{>}, s^+\big) \in C_f^+ \quad \mu\big(I^{>}\big) \leq s^+.$$

3. Pour tout $(I^{\geq}, s^-) \in C_f^-$ et tout $(I^{>}, s^+) \in C_f^+$

$$I^{\geq} \subseteq I^{>} \quad \Longrightarrow \quad s^- \leq s^+.$$

Démonstration. D'après le Corollaire 41, les assertions 1. et 2. sont équivalentes. On va donc prouver que 2 et 3 sont équivalentes. On commence par montrer que 2 implique 3. Supposons que 3 n'est pas vérifiée. Il existe donc $(I^{\geq}, s^-) \in C_f^-$ et $(I^{>}, s^+) \in C_f^+$ tels que $I^{\geq} \subseteq I^{>}$ et $s^- \not\leq s^+$. Dans ce cas, pour toute fonction $\mu : 2^{[n]} \to L$ vérifiant

$$s^- \leq \mu(I^{\geq}) \quad \text{et} \quad \mu(I^{>}) \leq s^+$$

on a $\mu(I^{\geq}) \not\leq \mu(I^{>})$, ce qui implique que μ n'est pas non décroissante. Donc la condition 2 n'est pas vérifiée.

On montre maintenant que 3 implique 2. Supposons que 3 est vérifiée. On définit la fonction $\mu : 2^{[n]} \to L$ par

$$\mu(I) = \bigvee \big\{ s^- \;\big|\; (I^{\geq}, s^-) \in C_f^- \text{ et } I^{\geq} \subseteq I \big\}.$$

Notez qu'on considère $\bigvee \emptyset = \bot$. La fonction ainsi définie est non décroissante (c'est donc une FEND), et pour tout $(I^{\geq}, s^-) \in C_f^-$ on a $s^- \leq \mu(I^{\geq})$.

Montrons maintenant que $\mu(I^{>}) \leq s^+$ est vérifiée pour tout $(I^{>}, s^+) \in C_f^+$. Soit $(I^{>}, s^+) \in C_f^+$. Puisque 3. est vérifiée on a

$$\forall (I^{\geq}, s^-) \in C_f^- : \quad I^{\geq} \subseteq I^{>} \Longrightarrow s^- \leq s^+.$$

Par conséquent

$$\mu(I^{>}) = \bigvee \big\{ s^- \;\big|\; (I^{\geq}, s^-) \in C_f^- \text{ et } I^{\geq} \subseteq I^{>} \big\} \leq s^+.$$

□

Pour finir, nous donnons une formulation explicite des bornes p^- et p^+ de $\mathcal{I}(f)$, en fonction de μ_{p^-} et de ν_{p^+}. Nous considérons ces deux fonctions d'ensemble en particulier (plutôt que μ_{p^+} et ν_{p^-}), car le nombre d'ensembles focaux de μ_{p^-} (resp. ν_{p^+}) est au plus égal au nombre de contraintes dans C_f^- (resp. C_f^+), respectivement. Ce point est détaillé dans la Remarque 44.

Proposition 43. Si $\mathcal{I}(f)$ est non vide, ses bornes inférieures et supérieures sont respectivement définies par

$$p^-(\mathbf{x}) = \bigvee_{I\subseteq[n]} \mu_{p^-}(I) \wedge \bigwedge_{i\in I} x_i \quad \text{et} \quad p^+(\mathbf{x}) = \bigwedge_{I\subseteq[n]} \nu_{p^+}(I) \vee \bigvee_{i\in I} x_i,$$

où les fonctions μ_{p^-} et ν_{p^+} sont données par

$$\mu_{p^-}(I) = \bigvee \left\{ s^- \;\middle|\; (I^{\geq}, s^-) \in C_f^- \quad \text{et} \quad I^{\geq} \subseteq I \right\} \tag{3.11}$$

et par

$$\nu_{p^+}(I) = \mu_{p^+}([n]\backslash I) = \bigwedge \left\{ s^+ \;\middle|\; (I^{>}, s^+) \in C_f^+ \quad \text{et} \quad [n]\backslash I \subseteq I^{>} \right\}. \tag{3.12}$$

Démonstration. La borne inférieure p^- de $\mathcal{I}(f)$ est la plus petite FLP telle que

$$\forall (I^{\geq}, s^-) \in C_f^-, \quad s^- \leq \mu_{p^-}(I). \tag{3.13}$$

Puisque μ_{p^-} est l'unique FEND qui spécifie une DNF de p^-, elle est la plus petite FEND qui vérifie (3.13). L'égalité (3.11) découle directement de cette observation. On exprime ν_{p^+} en fonction de μ_{p^+}. Cette dernière est la plus grande FEND qui vérifie

$$\forall (I^{>}, s^+) \in C_f^+, \quad \mu_{p^+}(I) \leq s^+.$$

□

Remarque 44. Notez que la fonction $\mu : 2^{[n]} \to L$ définie par

$$\mu(I) = \begin{cases} \bigvee \left\{ s^- \;\middle|\; (I^{\geq}, s^-) \in C_f^- \text{ et } I^{\geq} \subseteq I \right\} & \text{si } \exists s^- \in L \text{ tel que } (I, s^-) \in C_f^- \\ \bot & \text{sinon.} \end{cases}$$

se trouve dans l'intervalle $[\mu^*_{p^-}, \mu_{p^-}]$ et définit donc une DNF de p^-. Elle permet d'exprimer p^- en fonction d'un nombre de valeurs inférieur ou égal au nombre de contraintes dans C_f^-, qui est lui même inférieur ou égal à $|D|k$.

Afin de traiter la borne supérieure de $\mathcal{I}(f)$ de manière similaire, on considère la fonction $\nu : 2^{[n]} \to L$ définie par

$$\nu(I) = \begin{cases} \bigwedge \left\{ s^+ \;\middle|\; (I^{>}, s^+) \in C_f^+ \text{ et } [n]\backslash I \subseteq I^{>} \right\} & \text{si } \exists s^+ \in L \text{ tel que } ([n]\backslash I, s^+) \in C_f^+ \\ \top & \text{sinon.} \end{cases}$$

se trouve dans l'intervalle $[\nu_{p^+}, \nu^*_{p^+}]$ et définit donc une CNF de p^+.

Nous avons décrit le support théorique nécessaire à la définition d'un algorithme qui calcule les bornes de $\mathcal{I}(f)$ et vérifie si cet ensemble est vide en temps polynomial par rapport à $|D|$, n et k. Nous terminons cette section par un exemple illustrant les résultats que nous venons d'exposer.

	physique	littérature	économie	algèbre	général
étudiant 1	5	5	?	4	4
étudiant 2	3	3	3	3	3
étudiant 3	5	3	3	5	5
étudiant 4	?	1	2	4	3

TABLE 3.1 – Notes attribuées aux étudiants.

3.6.3 Exemple

Cette sous-section présente un cas fictif de modélisation de préférences multi-critères par une FLP, et la résolution du problème d'interpolation qui en découle.

Considérons des étudiants ayant été notés dans plusieurs cours, sur une échelle telle que

$$\text{insuffisant} < \text{passable} < \text{assez-bien} < \text{bien} < \text{très-bien}.$$

Pour des raisons de concision, ces évaluations sont notées respectivement 1, 2, 3, 4 et 5, et on appelle T la chaîne constituée de 1, 2, 3, 4 et 5. Un jury attribue une note générale à chaque étudiant, en fonction de ses notes dans les différents cours. La Table 3.1 contient les notes obtenues par quatre étudiants. Deux notes sont manquantes, certains étudiants ayant été dispensés de certains cours. Les notes manquantes sont indiquées par un point d'interrogation.

On cherche à modéliser par une fonction la manière dont le jury attribue une note générale à chaque étudiant en fonction de ses notes dans chaque matière. Afin de gérer les valeurs manquantes, on fait appel à un espace dont les éléments sont des sous-intervalles de $[1, 5]$, qui représentent chacun un état de connaissance partielle d'une valeur sur T. Une valeur $a \in T$ connue avec certitude est représentée par l'intervalle $[a, a]$, tandis qu'une valeur manquante est représentée par l'intervalle $[1, 5]$ (incertitude totale). On appelle $\mathbf{T}$ la famille des sous-intervalles de $[1, 5]$. On définit les opérations $\wedge$ et $\vee$ sur $\mathbf{T}$ par

$$[a, b] \vee [c, d] = [a \vee c, b \vee d] \quad \text{et} \quad [a, b] \wedge [c, d] = [a \wedge c, b \wedge d].$$

La définition des opérations $\wedge$ et $\vee$ du treillis $\mathbf{T}$ repose sur le principe suivant : pour tout $a, b \in \{1, 2, 3, 4, 5\}$, si $x \in [a, b]$ et $y \in [c, d]$, alors $x \wedge y \in [a \wedge c, b \wedge d]$ et $x \vee y \in [a \vee c, b \vee d]$. Les opérations $\wedge$ et $\vee$ sur $\mathbf{T}$ définissent un treillis distributif fini (voir Figure 3.3). On cherche à modéliser les décisions du jury par une FLP de $\mathbf{T}^4$ vers $\mathbf{T}$ compatible avec tous les exemples de la table. Si on représente la table par une fonction partielle f, alors les FLP compatibles avec tous les exemples de la table sont les FLP qui appartiennent à $\mathcal{I}(f)$.

Pour commencer, on définit un plongement de Dilworth de $\mathbf{T}$. Remarquez que le treillis $\mathbf{T}$ est isomorphe au sous-treillis de T^2 qui a pour univers le simplex d'ordre $\{(a, b) \in T^2 \mid a \leq b\}$. Soit $L_1 = L_2 = T$. On définit le plongement de Dilworth de $\mathbf{T}$ dans $L_1 \times L_2$ par

$$[a, b]_{|_1} = a \quad \text{et} \quad [a, b]_{|_2} = b,$$

pour tout $[a, b] \in \mathbf{T}$.

Pour rappel, chaque note $a \in T$ est représentée par l'intervalle $[a, a]$ et les valeurs manquantes sont représentées par l'intervalle $[1, 5]$. On dénote le tuple des évaluations locales d'un étudiant par $\mathbf{x}^i$, où i est le numéro de la ligne de l'étudiant dans la Table 3.1. Pour le premier étudiant, le modèle p doit vérifier

$$p(\mathbf{x}^1) = [4, 4] \quad \text{où } \mathbf{x}^1 = ([5, 5], [5, 5], [1, 5], [4, 4]).$$

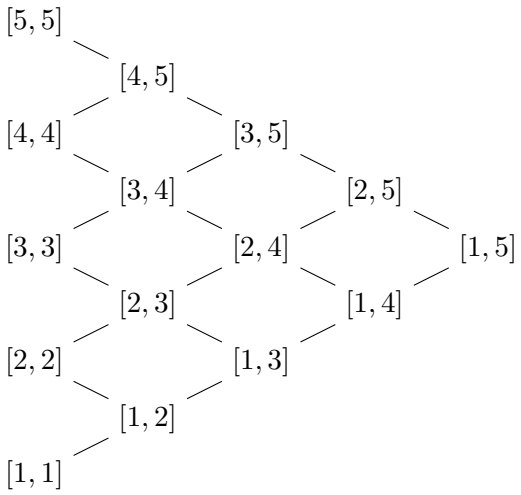


FIGURE 3.3 – Le treillis distributif $\mathbf{T}$ des subintervalles de $[1, 5]$.

On a

$$\begin{array}{llll} s^-_{[4,4],1} = [4,4] & s^+_{[4,4],1} = [4,5] & I^{\geq}_{f,\mathbf{x}^1,1} = \{1,2,4\} & I^{>}_{f,\mathbf{x}^1,1} = \{1,2\} \\ s^-_{[4,4],2} = [1,4] & s^+_{[4,4],2} = [4,4] & I^{\geq}_{f,\mathbf{x}^1,2} = \{1,2,3,4\} & I^{>}_{f,\mathbf{x}^1,2} = \{1,2,3\} \end{array}$$

et en appliquant la Proposition 39 on obtient

$$[4,4] \leq \mu_p(\{1,2,4\}) \quad \text{et} \quad \mu_p(\{1,2,3\}) \leq [4,4].$$

Pour le second étudiant, p doit vérifier

$$p(\mathbf{x}^2) = [3,3] \quad \text{où } \mathbf{x}^2 = ([3,3],[3,3],[3,3],[3,3]).$$

On a

$$\begin{array}{llll} s^-_{[3,3],1} = [3,3] & s^+_{[3,3],1} = [3,5] & I^{\geq}_{f,\mathbf{x}^2,1} = \{1,2,3,4\} & I^{>}_{f,\mathbf{x}^2,1} = \{\} \\ s^-_{[3,3],2} = [1,3] & s^+_{[3,3],2} = [3,3] & I^{\geq}_{f,\mathbf{x}^2,2} = \{1,2,3,4\} & I^{>}_{f,\mathbf{x}^2,2} = \{\} \end{array}$$

ainsi que les contraintes correspondantes :

$$[3,3] \leq \mu_p(\{1,2,3,4\}) \quad \text{et} \quad \mu_p(\{\}) \leq [3,3].$$

Pour le troisième étudiant, p doit vérifier

$$p(\mathbf{x}^3) = [5,5] \quad \text{où } \mathbf{x}^3 = ([5,5],[3,3],[3,3],[5,5]).$$

On a

$$\begin{array}{llll} s^-_{[5,5],1} = [5,5] & s^+_{[5,5],1} = [5,5] & I^{\geq}_{f,\mathbf{x}^3,1} = \{1,4\} & I^{>}_{f,\mathbf{x}^3,1} = \{\} \\ s^-_{[5,5],2} = [1,5] & s^+_{[5,5],2} = [5,5] & I^{\geq}_{f,\mathbf{x}^3,2} = \{1,4\} & I^{>}_{f,\mathbf{x}^3,2} = \{\} \end{array}$$

et les contraintes correspondantes :

$$[5,5] \leq \mu_p(\{1,4\}) \quad \text{et} \quad \mu_p(\{\}) \leq [5,5].$$

Pour le quatrième étudiant, p doit vérifier

$$p(\mathbf{x}^4) = [3,3] \quad \text{où } \mathbf{x}^4 = ([1,5],[1,1],[2,2],[4,4]).$$

On a

$$\begin{array}{llll} s^-_{[3,3],1} = [3,3] & s^+_{[3,3],1} = [3,5] & I^{\geq}_{f,\mathbf{x}^4,1} = \{4\} & I^{>}_{f,\mathbf{x}^4,1} = \{4\} \\ s^-_{[3,3],2} = [1,3] & s^+_{[3,3],2} = [3,3] & I^{\geq}_{f,\mathbf{x}^4,2} = \{1,4\} & I^{>}_{f,\mathbf{x}^4,2} = \{1,4\} \end{array}$$

et les contraintes correspondantes

$$[3,3] \leq \mu_p(\{4\}) \quad \text{et} \quad \mu_p(\{1,4\}) \leq [3,3].$$

Les contraintes caractérisant les interpolations de f sont représentées par les ensembles

$$\begin{aligned} C_f^- &= \Big\{(\{1,2,4\},[4,4]),\ (\{1,2,3,4\},[3,3]),\ (\{1,4\},[5,5]),\ (\{4\},[3,3])\Big\}, \\ C_f^+ &= \Big\{(\{1,2,3\},[4,4]),\ (\{\},[3,3]),\ (\{\},[5,5]),\ (\{1,4\},[3,3])\Big\}. \end{aligned}$$

Remarquez qu'aucune FLP n'est à la fois compatible avec la contrainte $(\{1,4\},[5,5])$ de C_f^- et la contrainte $(\{1,4\},[3,3])$ de C_f^+, puisque $\{1,4\} \subseteq \{1,4\}$ et $[5,5] > [3,3]$. En effet, aucune FEND $\mu : 2^{[n]} \to L$ ne peut vérifier

$$[5,5] \leq \mu(\{1,4\}) \quad \text{et} \quad \mu(\{1,4\}) \leq [3,3].$$

En d'autres termes, aucune FLP n'est compatible simultanément avec les lignes 3 et 4 de la Table 3.1. Supposons qu'on décide de supprimer la 4-ème ligne de la Table. On obtient alors les ensembles de contraintes

$$\begin{aligned} C_f^- &= \Big\{(\{1,2,4\},[4,4]),\ (\{1,2,3,4\},[3,3]),\ (\{1,4\},[5,5])\Big\}, \\ C_f^+ &= \Big\{(\{1,2,3\},[4,4]),\ (\{\},[3,3]),\ (\{\},[5,5])\Big\}. \end{aligned}$$

Ces ensembles contiennent des informations redondantes ; certains couples (marqués en rouge) peuvent être supprimés sans modifier l'ensemble de FLP caractérisé par les contraintes. Par exemple, dans l'ensemble C_f^+, le couple $(\{1,2,3\},[4,4])$ représente une contrainte plus forte que $(\{\},[5,5])$, puisque toute FEND μ qui vérifie $\mu(\{1,2,3\}) \leq [4,4]$ vérifie également $\mu(\{\}) \leq [5,5]$. On considère les ensembles de contraintes simplifiés C_f^{-*} et C_f^{+*} :

$$\begin{aligned} C_f^{-*} &= \Big\{(\{1,4\},[5,5])\Big\}, \\ C_f^{+*} &= \Big\{(\{1,2,3\},[4,4]),\ (\{\},[3,3])\Big\}. \end{aligned}$$

On dénote par p^- et p^+ les bornes (respectivement, inférieure et supérieure) de $\mathcal{I}(f)$. En s'appuyant sur la Proposition 43 et la Remarque 44, on exprime les bornes p^- et p^+ de $\mathcal{I}(f)$ en fonction de C_f^{-*} et C_f^{+*}. La fonction $\mu^*_{p^-}$ est donnée par

$$\mu^*_{p^-}(I) = \begin{cases} [5,5] \text{ si } I = \{1,4\}, \\ \bot \text{ sinon.} \end{cases}$$

Donc p^- est la fonction définie par

$$p^-(\mathbf{x}) = [5,5] \wedge x_1 \wedge x_4.$$

Pour obtenir p^+, on exprime $\nu^*_{p^+}$ d'après les contraintes de C_f^{+*}. La fonction $\nu^*_{p^+}$ est donnée par

$$\nu^*_{p^+}(I) = \begin{cases} [3,3] \text{ si } [n]\backslash I = \{\} \text{ (c.à.d. } I = \{1,2,3,4\}), \\ [4,4] \text{ si } [n]\backslash I = \{1,2,3\} \text{ (c.à.d. } I = \{4\}), \\ \top \text{ sinon.} \end{cases}$$

Donc p^+ est la fonction définie par

$$p^+(\mathbf{x}) = \Big([3,3] \vee x_1 \vee x_2 \vee x_3 \vee x_4\Big) \wedge \Big([4,4] \vee x_4\Big).$$

3.7 Calcul des bornes de $\mathcal{I}(f)$ en temps polynomial

3.7.1 Algorithmes

Les résultats de la section précédente permettent de définir des algorithmes qui résolvent le problème d'interpolation en temps polynomial par rapport à n, $|D|$ et k. L'Algorithme 9 définit une fonction qui construit et renvoie des dictionnaires représentant les ensembles C_f^- et C_f^+ pour une fonction f donnée. L'Algorithme 12 définit une fonction qui teste si l'ensemble des FLP décrites par C_f^- et C_f^+ est vide ou non.

Dans les Algorithmes 9, 10 et 11, C_f^- et C_f^+ sont représentés par les dictionnaires $\mathtt{C}^-$ et $\mathtt{C}^+$, dont les clés sont des sous-ensemble de $[n]$ et les valeurs sont des éléments de L. La valeur associée à une clé G dans $\mathtt{C}^-$ ou $\mathtt{C}^+$ est dénotée par $\mathtt{C}^-[G]$ ou $\mathtt{C}^+[G]$, respectivement.

Algorithme 9: Calcule les dictionnaires $\mathtt{C}^-$ et $\mathtt{C}^+$, étant donné une fonction $f : D \to L$.

```
1  fonction CALCULER_CONTRAINTES(f)
2      C⁻ ← dictionnaire vide
3      C⁺ ← dictionnaire vide
4      pour x ∈ D faire
5          pour t ∈ {1, ..., k} faire
6              I^≥ ← {i | x_i|_t ≥ f(x)|_t}
7              I^> ← {i | x_i|_t > f(x)|_t}
8              s⁻ ← ⋀{a ∈ L | a|_t = f(x)|_t}
9              s⁺ ← ⋁{a ∈ L | a|_t = f(x)|_t}
10             MAJ⁻(C⁻, I^≥, s⁻)
11             MAJ⁺(C⁺, I^>, s⁺)
12     retourner C⁻, C⁺
```

À chaque itération des boucles **pour**, une contrainte minimale et une contrainte maximale caractérisant les interpolations de f sont calculées et ajoutées à C_f^+ et C_f^- via fonctions UPDATE$^-$

et UPDATE$^+$, respectivement. Ces deux fonctions sont décrites par les Algorithmes 10 et 11.

Algorithme 10: Mise à jour des contraintes minimales.

```
1 fonction MAJ⁻(C⁻, I≥, s⁻)
2     si I≥ est une clé dans C⁻ alors
3         C⁻[I≥] ← C⁻[I≥] ∨ s⁻
4     alors
5         C⁻[I≥] ← s⁻
```

Algorithme 11: Mise à jour des contraintes maximales.

```
1 fonction MAJ⁺(C⁺, I>, s⁺)
2     si I> est une clé dans C⁺ alors
3         C⁺[I>] ← C⁺[I>] ∧ s⁺
4     alors
5         C⁺[I>] ← s⁺
```

Remarque 45. Lorsqu'on a deux contraintes minimales

$$\mu(I^{\geq}) \geq s_1^- \quad \text{et} \quad \mu(I^{\geq}) \geq s_2^-,$$

exprimées par rapport à un même ensemble $I^{\geq}$, celles-ci peuvent être résumées en une seule contrainte équivalente $\mu(I^{\geq}) \geq s_1^- \vee s_2^-$. De manière duale, deux contraintes minimales

$$\mu(I^{>}) \leq s_1^+ \quad \text{et} \quad \mu(I^{>}) \leq s_2^+,$$

exprimées par rapport à un même ensemble $I^{>}$, peuvent être résumées en une seule contrainte équivalente $\mu(I^{>}) \leq s_1^+ \wedge s_2^+$. Par conséquent, lorsqu'un couple (clé,valeur) dont la clé appartient déjà à la table $\mathtt{C}^-$ (resp. $\mathtt{C}^+$) y est insérée (voir Algorithmes 10 et 11, ligne 2), l'ancienne et la nouvelle valeur associées à la clé sont simplement combinées par l'opération $\vee$ (resp. $\wedge$).

Enfin, l'Algorithme 12 s'appuie sur la Proposition 42 pour tester s'il existe une FLP qui satisfait toutes les contraintes de C_f^- et C_f^+. La fonction TEST_CONTRAINTES renvoie vrai si une telle FLP existe, faux sinon.

Algorithme 12: Renvoie vrai si et seulement si les contraintes données par $\mathtt{C}^-$ et $\mathtt{C}^+$ caractérisent un ensemble non vide.

```
1 fonction TEST_CONTRAINTES(C⁻, C⁺)
2     pour chaque (I>, s⁺) dans C⁺ faire
3         pour chaque (I≥, s⁻) dans C⁻ faire
4             si I≥ ⊆ I> et s⁻ ⋢ s⁺ alors
5                 retourner faux
6     retourner vrai
```

3.7.2 Complexité

Le temps d'exécution de l'algorithme dépend en partie du temps d'exécution des opérations $\wedge$ et $\vee$, et donc de l'implémentation de L. On suppose que chaque élément de L est représenté

par un tuple de valeurs entières, comme décrit dans la Sous-section 1.6.2. Cependant, dans certains cas il est intéressant de coupler cette représentation à une représentation plus rapide de L (par exemple un représentation par des tableaux, comme décrit dans la Sous-section 1.6.1). On exprimera donc la complexité des algorithmes en fonction d'une inconnue Δ, où $O(\Delta)$ est la complexité au pire cas des tâches suivantes : tester la condition $a \leq b$, retourner $a \wedge b$ et retourner $a \vee b$. Enfin, on suppose que, étant donné $\mathbf{x} \in D$, on peut obtenir $f(\mathbf{x})$ en $O(1)$.

Proposition 46. On peut implémenter l'Algorithme 9 de telle sorte que la fonction CALCULER_CONTRAINTES s'exécute en $O(|D|k(n + |L|\Delta))$.

Démonstration. Les lignes 6 et 7 de l'algorithme s'exécutent en $O(n)$ (on réalise n comparaisons entre des nombres entiers).

Calculer la valeur de $s^-_{f(\mathbf{x}),t}$ et $s^+_{f(\mathbf{x}),t}$ (lignes 8 et 9) pour des valeurs de $f(\mathbf{x})$ et t données peut être fait en $O(|L|\Delta)$.

Il reste à évaluer la complexité de MAJ$^-$ et MAJ$^+$. En faisant appel à une implémentation appropriée des dictionnaires $\mathtt{C}^-$ et $\mathtt{C}^+$ (par exemple, un trie [14], Section 8.1), on peut garantir qu'une recherche ou une insertion dans la table $\mathtt{C}^-$ ou $\mathtt{C}^+$ s'effectuera en $O(n)$. La complexité des fonctions MAJ$^-$ et MAJ$^+$ est donc $O(n + \Delta)$.

Les lignes 6 à 11 de l'Algorithme 9 sont exécutées $|D|k$ fois, donc complexité de CALCULER_CONTRAINTES est

$$O\big(|D|k(n + |L|\Delta + n + \Delta)\big) = O\big(|D|k(n + |L|\Delta)\big).$$

□

Remarque 47. Les valeurs de $s^-_{f(\mathbf{x}),t}$ et $s^+_{f(\mathbf{x}),t}$ étant déterminées par $f(\mathbf{x})_{|_t}$, elles ont respectivement $|L_1| + \cdots + |L_k|$ valeurs possibles. Afin que le calcul de $s^-_{f(\mathbf{x}),t}$ et $s^+_{f(\mathbf{x}),t}$ n'impacte pas la complexité globale de l'algorithme, leurs valeurs peuvent être pré-calculées et stockées dans un tableau de taille $O(|L_1| + \cdots + |L_k|)$. Ce pré-calcul s'effectue en

$$O((|L_1| + \cdots + |L_k|)|L|\Delta) = O(|\mathcal{J}(L)||L|\Delta).$$

La complexité de CALCULER_CONTRAINTES devient alors $O(|D|k(n + \Delta))$.

Proposition 48. On peut implémenter l'Algorithme 12 de telle sorte que la fonction TEST_CONTRAINTES s'exécute en $O(|D|^2k^2(n + \Delta))$.

Démonstration. Puisque le nombre de contraintes stockées dans $\mathtt{C}^-$ et $\mathtt{C}^+$ est au maximum $|D|k$, la ligne 4 de l'Algorithme 12 s'exécute au plus $|D|^2k^2$ fois. On peut tester $I^{\geq} \subseteq I^{>}$ en $O(n)$ et $s^- \not\leq s^+$ en $O(\Delta)$. □

Enfin, les dictionnaires $\mathtt{C}^-$ et $\mathtt{C}^+$ issus de la fonction CALCULER_CONTRAINTES constituent des représentation en mémoire des bornes p^- et p^+ de $\mathcal{I}(f)$ (si $\mathcal{I}(f) \neq \emptyset$). Le nombre de clés dans chacun de ces dictionnaires étant au plus $|D|k$, les Algorithmes 7 et 8 permettent respectivement de calculer $p^-(\mathbf{x})$ et $p^+(\mathbf{x})$ en $O(|D|kn\Delta)$, pour tout $\mathbf{x} \in D$.

Notez que notre méthode se décompose en deux grandes étapes. Dans un premier temps on calcule des contraintes qui caractérisent les bornes de $\mathcal{I}(f)$ (dans le cas où $\mathcal{I}(f) \neq \emptyset$). Dans un deuxième temps, on vérifie si les FLP p^- et p^+ caractérisées par ces contraintes décrivent un intervalle (c'est à dire si $p^- \leq p^+$). Ce schéma est similaire à celui de la méthode proposée dans [24] où les deux FLP p^- et p^+ (respectivement dénotées par p^{cl} et p^{int} dans la Sous-section 3.5.1) sont calculées, avant de tester si $p^- \leq p^+$.

Notre méthode se différentie principalement de celle de [24], en faisant appel à un nombre réduit de contraintes afin de décrire les bornes de $\mathcal{I}(f)$.

De ce fait, la première étape est effectuée en $O(|D|k(n+|L|\Delta))$ par notre méthode, tandis qu'elle est effectuée en $O(2^n|D|n\Delta)$ par la méthode de [24] (voir Remarque 34). De plus, le temps nécessaire pour tester si $p^- \leq p^+$ est $O(|D|^2k^2(n+\Delta))$ dans notre méthode, contre $O(2^n\Delta)$ dans la méthode de [24].

3.7.3 Génération de données pour l'étude du temps d'exécution

Les Propositions 46 et 48 donnent une borne supérieur de l'augmentation du temps d'exécution au pire cas des Algorithmes 9 et 12, en fonction des dimensions du problème. Cependant, il se peut qu'en pratique le temps d'exécution moyen des algorithmes soit moins important.

Afin d'évaluer empiriquement ce temps d'exécution moyen, nous avons réalisé des tests à partir de données générées aléatoirement. Cette solution permet de réaliser des tests sur des données de dimensions variables.

On génère aléatoirement une fonction partielle $f : D \to L$ (où $D \subseteq L^n$) en suivant les trois étapes ci-dessous (qui sont détaillées plus loin).

A. Génération aléatoire d'un treillis distributif fini L.

B. Génération aléatoire d'une FLP $p : L^n \to L$,

C. Génération aléatoire d'un ensemble $D \subseteq L^n$.

La fonction $f : D \to L$ est définie par $f(\mathbf{x}) = p(\mathbf{x})$ pour tout $\mathbf{x} \in D$.

A. Génération aléatoire d'un treillis distributif fini

Chaque treillis distributif fini est caractérisé par l'ensemble ordonné de ses éléments join-irréductibles (voir Sous-section 1.5.2). Par conséquent, la génération aléatoire d'un treillis distributif fini peut être vue comme la génération aléatoire d'un ensemble partiellement ordonné. On choisit de considérer la hauteur du treillis comme un paramètre de la méthode de génération (c'est à dire que celle-ci sera décidée a priori). La hauteur d'un treillis distributif fini correspond au nombre de ses éléments join-irréductibles. Par conséquent, la tâche de génération aléatoire d'un treillis de hauteur donnée peut être reformulée de la manière suivante : pour un entier h donné et un ensemble X de taille h, générer aléatoirement un ordre partiel $\leq$ sur X. Cette tâche peut être réalisée de plusieurs manières [49]. Nous utilisons la méthode décrite ci-dessous, qui dépend d'un second paramètre $\rho \in [0, 1]$.

Procédure de génération d'un treillis distributif fini(h, ρ)**.**

1. *Soit* $X = \{x_1, \ldots, x_h\}$ *et* $R = \{(x_1, x_1), \ldots, (x_h, x_h)\}$.
2. *Pour tout* $i, j \in [h]$ *tels que* $i < j$*, on ajoute* (x_i, x_j) *à* R *avec une probabilité* ρ.
3. *L'ensemble* R *étant une relation réflexive et acyclique sur* X*, sa fermeture transitive* $\leq$ *est un ordre partiel sur* X.
4. *On retourne le treillis* $(\mathcal{O}(X), \subseteq)$.

Notez que le paramètre ρ régule la "linéarité" de l'ordre partiel obtenu ; plus la valeur de ρ est élevée, plus la probabilité que deux éléments du treillis tirés au hasard soient comparable augmente ; lorsque $\rho = 0$, la relation $\leq$ obtenue est toujours une relation d'égalité (c'est à dire que $x_i \leq x_j \Rightarrow i = j$), et $(\mathcal{O}(X), \subseteq)$ est l'ensemble des parties de X ordonnées par inclusion ; lorsque $\rho = 1$, $\leq$ est un ordre total sur X, et $(\mathcal{O}(X), \subseteq)$ est une chaîne.

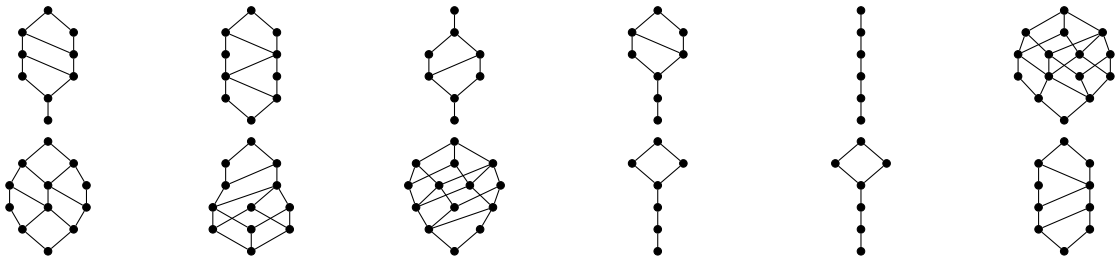


FIGURE 3.4 – Échantillon de 10 treillis distributifs générés aléatoirement ($h = 5$, $\rho = 0.6$).

B. Génération aléatoire d'une FLP

On génère une FLP p en spécifiant la valeur de μ_p sur chaque sous-ensemble de $[n]$. Cette procédure dépend d'un paramètre g qui détermine le nombre maximal d'ensembles focaux de μ_p.

1. On tire aléatoirement g sous-ensembles de $[n]$ (selon une probabilité uniforme sur $2^{[n]}$). On dénote l'ensemble des sous-ensembles ainsi tirés par $\mathbf{F} = \{I_1, \ldots, I_g\}$. La fonction μ_p sera définie de telle sorte que tous ses ensembles focaux sont dans $\mathbf{F}$.
2. Pour i de 1 à g, on pose $\mu_p(I_i) = a_i$ où a_i est un élément de l'intervalle $[a^-, a^+]$, tiré aléatoirement selon une loi de probabilité uniforme, dont les bornes sont définies par

$$a^- = \bigvee \Big\{ \mu_p(I_j) \;\Big|\; j \in [1, i-1] \text{ et } I_j \subset I_i \Big\} \quad \text{et} \quad a^+ = \bigwedge \Big\{ \mu_p(I_j) \;\Big|\; j \in [1, i-1] \text{ et } I_i \subset I_j \Big\}.$$

   Notez que a_1 est tiré dans l'intervalle $[\bot, \top]$. Pour tout $J \in 2^{[n]} \backslash \mathbf{F}$, on pose

$$\mu_p(J) = \bigvee \Big\{ \mu_p(I_i) \;\Big|\; I_i \in \mathbf{F} \text{ et } \; I_i \subseteq J \Big\},$$

   ce qui implique que J est un ensemble non focal de μ_p (la valeur de μ_p sur J n'a donc pas à être mémorisée, voir Sections 3.2 et 3.3).

Cette méthode de génération garantit que μ_p est une FEND possédant au plus g ensembles focaux.

C. Génération aléatoire d'un ensemble $D \subseteq L^n$

Soit m la taille choisie pour le domaine de f. On tire aléatoirement m éléments de L^n selon une loi de probabilité uniforme. On dénote l'ensemble de ces éléments par

$$D = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\}.$$

La fonction partielle $f : D \to L$ est définie par $f(\mathbf{x}) = p(\mathbf{x})$.

En tout, cette procédure de génération aléatoire dépend donc des 5 paramètres h, ρ, n, g et m.

3.7.4 Étude du temps d'exécution

Nous avons implémenté les algorithmes en Java [3]. Les tests ont été effectués sur un CPU Intel Core i7-4600U. L'implémentation des algorithmes représente le treillis L à l'aide d'un plongement de Dilworth trivial (autrement dit, pour tout treillis on a $k = |\mathcal{J}(L)|$, et les éléments du treillis sont représentés par des tuples binaires).

Pour rappel, la génération des données dépend de cinq paramètres :

3. `https://github.com/QGBrabant/javaggregation`

		Nombre maximal d'ensembles focaux (g)					
		10	50	100	200	500	1000
ρ	0.0	151.0	159.6	147.0	171.5	142.3	147.8
	0.2	153.2	149.1	132.3	135.5	132.9	146.8
	0.4	134.1	139.7	161.2	139.4	136.7	136.3
	0.6	147.3	140.2	131.3	154.2	138.5	132.2
	0.8	130.2	135.0	140.4	132.5	132.4	132.9
	1.0	133.6	137.8	125.8	127.1	137.1	139.3

TABLE 3.2 – Temps d'exécution des Algorithmes 9 et 12 (en millisecondes) en fonction du nombre maximal g d'ensembles focaux et de la probabilité ρ (avec $n = 30$, $m = 10000$, $h = 10$ et $p = 0.6$).

- h : la hauteur du treillis considéré, c'est à dire son nombre d'éléments join-irréductibles,
- ρ : une valeur de probabilité qui influence la génération aléatoire du treillis,
- n : l'arité de la fonction f,
- g : le nombre maximal d'ensemble focaux de FLP ayant servi à générer f,
- m : la taille du domaine de f.

Afin de tenter d'isoler l'effet de chaque paramètre h, ρ, n, g et m sur le temps d'exécution de nos algorithmes, on lance plusieurs tests.

Un test se déroule en deux étapes :

- génération aléatoire de f,
- mesure du temps d'exécution de 9 et 12.

Nous avons réalisé une première série de test, en faisant varier uniquement les valeurs de ρ et g. La Table 3.2 donne, pour chaque valeur de ρ et g, le temps d'exécution obtenu (moyenné sur 10 tests). On constate que ces deux paramètres ont peu d'impact sur le temps d'exécution. La Table 3.3 donnes les temps d'exécution pour des valeurs fixes de ρ et g et des valeurs de n, $|D|$, h variables. Les temps d'exécution présentés dans la Table 3.3 sont cohérents avec la complexité au pire cas $O(|D|^2k^2(n + \Delta))$ calculée précédemment. De plus, on constate que l'augmentation du temps d'exécution moyen est plus faible que ce que cette complexité au pire cas aurait suggéré. On constate que ce temps augmente, approximativement, linéairement par rapport à h (pour rappel, dans ces expériences on a toujours $h = k$), et légèrement plus que linéairement par rapport à m et n, tout en étant peu influencé par ρ and g.

Remarque 49. Nous n'avons pas observé d'influence du paramètre ρ sur le temps d'exécution. Cependant le fait que notre implémentation fasse usage de plongements de Dilworth triviaux uniquement force l'égalité $k = |\mathcal{J}(L)|$. Dans une situation où k est potentiellement inférieur à la dimension $|\mathcal{J}(L)|$ du treillis, on pourrait s'attendre à ce qu'une grande valeur de ρ favorise une plus petite valeur k (plus un grand nombre de paires d'éléments du treillis sont comparables entre eux, plus le nombre de chaînes nécessaires à partitionner l'ensemble $\mathcal{J}(L)$ tend à diminuer), et donc qu'une valeur élevée de ρ fasse diminuer le temps de calcul.

Arité (n) \ Taille du domaine (m)	10	100	1000	10000	50000
1	0.2	0.1	0.3	1.9	10.8
2	0.0	0.0	0.3	2.5	15.1
3	0.0	0.0	0.3	3.1	17.6
4	0.0	0.1	0.3	3.4	21.8
5	0.0	0.1	0.4	4.1	25.0
6	0.0	0.1	0.4	5.4	25.7
10	0.0	0.1	0.7	6.7	31.8
15	0.0	0.1	0.7	8.2	44.3
20	0.0	0.2	1.1	9.5	50.5
30	0.0	0.2	1.3	13.4	66.3

$h = 1$

Arité (n) \ Taille du domaine (m)	10	100	1000	10000	50000
1	0.1	0.2	0.5	3.4	15.7
2	0.0	0.1	0.5	4.3	23.6
3	0.0	0.1	0.5	5.1	29.0
4	0.0	0.1	0.6	6.5	32.9
5	0.0	0.1	0.7	6.9	36.7
6	0.0	0.1	0.8	8.4	38.6
10	0.0	0.2	1.2	11.7	60.8
15	0.0	0.2	1.4	16.2	91.2
20	0.0	0.2	1.8	19.3	96.6
30	0.1	0.3	2.6	24.3	134.4

$h = 2$

Arité (n) \ Taille du domaine (m)	10	100	1000	10000	50000
1	0.1	0.3	1.0	7.7	34.7
2	0.0	0.1	1.2	9.8	45.1
3	0.0	0.1	1.1	11.0	55.8
4	0.0	0.2	1.4	12.9	64.4
5	0.0	0.2	1.7	16.6	76.9
6	0.1	0.2	1.7	17.3	87.3
10	0.1	0.3	3.2	25.9	133.0
15	0.1	0.3	3.6	39.6	215.1
20	0.1	0.5	3.9	46.1	250.8
30	0.1	0.7	5.6	60.5	334.1

$h = 5$

Arité (n) \ Taille du domaine (m)	10	100	1000	10000	50000
1	0.1	0.4	1.5	13.8	62.2
2	0.0	0.2	2.0	19.3	82.2
3	0.0	0.2	2.2	19.9	98.6
4	0.1	0.3	2.6	23.3	121.0
5	0.1	0.4	3.3	29.5	143.8
6	0.1	0.4	3.6	32.5	159.0
10	0.1	0.7	5.5	50.6	249.7
15	0.1	0.7	6.9	73.4	396.7
20	0.1	0.9	9.3	93.4	510
30	0.2	1.3	13.0	131.3	696.3

$h = 10$

Arité (n) \ Taille du domaine (m)	10	100	1000	10000	50000
1	0.1	0.4	2.5	22.6	113.9
2	0.1	0.4	3.3	29.7	148.9
3	0.1	0.4	3.6	37.4	188.6
4	0.1	0.5	5.0	44.9	228.7
5	0.1	0.7	6.4	54.8	271.9
6	0.1	0.7	6.5	59.2	306.6
10	0.2	1.1	10.8	96.0	485.6
15	0.2	1.4	13.5	145.1	769.2
20	0.2	1.8	16.3	180.6	995.5
30	0.3	2.4	24.2	249.9	1387.8

$h = 20$

Arité (n) \ Taille du domaine (m)	10	100	1000	10000	50000
1	0.2	0.6	3.6	32.8	166.9
2	0.1	0.6	5.0	44.4	217.8
3	0.1	0.6	5.9	55.6	279.9
4	0.1	0.8	7.0	68.1	333.4
5	0.1	0.9	8.4	85.1	401.4
6	0.2	1.0	9.7	89.4	444.9
10	0.2	1.8	14.4	137.3	691.6
15	0.2	1.9	21.2	213.2	1099.7
20	0.4	2.6	26.5	266.4	1435.6
30	0.4	3.6	38.1	374.4	2323.8

$h = 30$

TABLE 3.3 – Temps d'exécution (en millisecondes), pour des valeurs variables de n, h, et m ($g = 200$, $\rho = 0.6$).

3.8 Conclusion

Dans ce chapitre, nous avons généralisé une approche existante afin de calculer les bornes de $\mathcal{I}(f)$, pour une fonction partielle $f : D \to L$ (où $D \subseteq L^n$) lorsque L est un treillis distributif borné qui peut être plongé dans un produit fini de chaînes. Ce calcul et la vérification que $\mathcal{I}(f)$ est non-vide s'effectue en temps polynomial par rapport aux dimensions du problème (qui sont : l'arité de f, la taille de son domaine et la taille du treillis L).

Cette méthode a pour résultat deux ensembles C_f^- et C_f^+, qui peuvent être interprétés respectivement comme la description de la plus petite et de la plus grande FEND μ qui spécifie une interpolation de f.

Des tests effectués sur des données générées aléatoirement montrent que cette méthode peut être appliquée pour calculer les bornes de $\mathcal{I}(f)$ lorsque les dimensions du problème sont de taille conséquente.

Chapitre 4

Fonctions latticielles polynomiales k-maxitives

Nous terminons cette partie par l'étude théorique d'une sous-classe de FLP sur les treillis distributifs, qu'on appelle FLP de degré k.

4.1 Préliminaire

L'étude des FLP de degré k est motivée par l'analogie entre les intégrales de Sugeno (k-maxitives) et les intégrales de Choquet (k-additives).

Soit $\mu : 2^{[n]} \to [0,1]$ une capacité. L'intégrale de Choquet $C_\mu : [0,1]^n \to [0,1]$ est la fonction définie par

$$C_\mu(\mathbf{x}) = \sum_{I \subseteq [n]} m_\mu(I) * \bigwedge_{i \in I} x_i,$$

où m_μ est la transformée de Möbius de μ (voir Chapitre 5). On dit que μ est k-additive si

$$m_\mu(I) = 0$$

pour tout I de cardinalité supérieure à k. Une intégrale de Choquet est dite k-additive si elle est définie par rapport à une capacité k-additive.

Certaines études montrent que l'utilisation d'intégrales de Choquet k-additives (où $k < n$) permet de modéliser des données empiriques (dans le cadre de l'apprentissage de préférences et de la classification) avec une précision similaire, voire supérieure, à celle obtenue par des intégrales de Choquet non restreintes [95, 94]

Soit un treillis distributif borné L et une capacité $\mu : 2^{[n]} \to L$. L'intégrale de Sugeno $S_\mu : L^n \to L$ est la fonction définie par

$$S_\mu(\mathbf{x}) = \bigvee_{I \subseteq [n]} \mu(I) \wedge \bigwedge_{i \in I} x_i.$$

On dit que μ est k-maxitive si, pour tout $I \subseteq [n]$,

$$|I| > k \quad \Longrightarrow \quad \mu(I) = \bigvee_{J \subset I} \mu(J).$$

On dit que S_μ est k-maxitive si μ est k-maxitive.

On appelle *terme* toute expression composée de variables et de l'opérateur $\wedge$, autrement dit toute expression de la forme

$$\bigwedge_{i\in I} x_i,$$

où $I \subseteq [n]$. Une intégrale de Choquet k-additive peut être exprimée par une somme de termes pondérés de tailles au plus k, tandis qu'une intégrale de Sugeno k-maxitive peut être exprimée par un supremum de termes de tailles au plus k précédés d'une constante. En ce sens la k-maxitivité (pour les intégrales de Sugeno) peut être vue comme une propriété analogue à la k-additivité (pour l'intégrale de Choquet), bien que les études concernant les applications des intégrales de Sugeno k-maxitives [13, 99] soient trop préliminaires pour permettre de conclure que ces intégrales restreintes puissent jouer un rôle similaire aux intégrales de Choquet k-additives dans les domaines de la modélisation de préférence ou de la classification.

Dans le reste de ce chapitre, nous étendons la notion de k-maxitivité aux FEND et aux FLP, et nous présentons une caractérisation des FLP k-maxitives.

4.2 Classes de FLP k-maxitives

Soit un treillis distributif L. Une FEND $\mu : 2^{[n]} \to L$ est *k-maxitive* si, pour tout $I \subseteq [n]$,

$$|I| > k \quad \Longrightarrow \quad \mu(I) = \bigvee_{J\subset I} \mu(J). \tag{4.1}$$

De cette définition, il découle qu'une FEND k-maxitive est entièrement déterminée par sa valeur sur des ensembles de taille au plus k. Une FLP p est dite k-maxitive si la fonction μ_p est k-maxitive.

Remarque 50. Les FEND 1-maxitives correspondent aux mesures de possibilité. Les FEND k-maxitives peuvent donc être vues comme des généralisations des mesures de possibilité, et sont parfois appelées *k-order possibility measures* [79].

On appelle *degré* d'une FLP $p \in \mathcal{P}_L^{(n)}$. la valeur minimale de k telle que p est k-maxitive.

Remarque 51. Cette définition généralise la définition de degré pour les fonctions booléennes croissantes, donnée dans [32].

On appelle $\mathcal{K}_L^{(k|n)}$ la classe des FLP k-maxitive sur L.

Remarque 52. On a

$$\mathcal{K}_L^{(0|n)} \subset \mathcal{K}_L^{(1|n)} \subset \cdots \subset \mathcal{K}_L^{(n|n)} = \mathcal{P}_L^{(n)}.$$

De plus, $\mathcal{K}_L^{(0|n)}$ est la classe des fonctions constantes, et $\mathcal{K}_L^{(1|n)}$ correspond à la classe des *weighted supremum functions* décrite dans [26].

Exemple 53. Soit L l'interval entier [0,3], et μ_p, μ_q deux capacités de $2^{\{1,2,3\}}$ vers L définies par

$$\begin{array}{llll}
\mu_p(\emptyset) = 1, & \mu_p(\{1,2\}) = 2, & \mu_q(\emptyset) = 0, & \mu_q(\{1,2\}) = 1,\\
\mu_p(\{1\}) = 2, & \mu_p(\{1,3\}) = 2, & \mu_q(\{1\}) = 0, & \mu_q(\{1,3\}) = 1,\\
\mu_p(\{2\}) = 2, & \mu_p(\{2,3\}) = 2, & \mu_q(\{2\}) = 1, & \mu_q(\{2,3\}) = 2,\\
\mu_p(\{3\}) = 1, & \mu_p(\{1,2,3\}) = 2, & \mu_q(\{3\}) = 1, & \mu_q(\{1,2,3\}) = 3.
\end{array}$$

On a donc

$$\begin{array}{llll}
\mu_p^*(\emptyset) = 1, & \mu_p^*(\{1,2\}) = 0, & \mu_q^*(\emptyset) = 0, & \mu_q^*(\{1,2\}) = 0,\\
\mu_p^*(\{1\}) = 2, & \mu_p^*(\{1,3\}) = 0, & \mu_q^*(\{1\}) = 0, & \mu_q^*(\{1,3\}) = 0,\\
\mu_p^*(\{2\}) = 2, & \mu_p^*(\{2,3\}) = 0, & \mu_q^*(\{2\}) = 1, & \mu_q^*(\{2,3\}) = 2,\\
\mu_p^*(\{3\}) = 0, & \mu_p^*(\{1,2,3\}) = 0, & \mu_q^*(\{3\}) = 1, & \mu_q^*(\{1,2,3\}) = 3.
\end{array}$$

Les FLP associées sont

$$\begin{aligned}
p(x_1,x_2,x_3) &= 1 \vee (2 \wedge x_1) \vee (2 \wedge x_2),\\
q(x_1,x_2,x_3) &= (1 \wedge x_2) \vee (1 \wedge x_3) \vee (2 \wedge x_2 \wedge x_3) \vee (3 \wedge x_1 \wedge x_2 \wedge x_3).
\end{aligned}$$

La fonction μ_q est 3-maxitive (et pas 2-maxitive), tandis que la fonction μ_p est 1-maxitive (et pas 0-maxitive). Les FLP p et q sont donc respectivement de degré 1 et 3 ; q appartient à $\mathcal{K}_L^{(3|n)}$ mais pas à $\mathcal{K}_L^{(1|n)}$ ni à $\mathcal{K}_L^{(2|n)}$, tandis que p appartient à $\mathcal{K}_L^{(1|n)}$, $\mathcal{K}_L^{(2|n)}$ et $\mathcal{K}_L^{(3|n)}$.

4.3 Caractérisation des FLP k-maxitives

Les FLP k-maxitives peuvent être caractérisées de la manière suivante.

Théorème 54. Pour tout $p \in \mathcal{P}_L^{(n)}$ et tout entier positif k, les trois propositions suivantes sont équivalentes.

(i) $p \in \mathcal{K}_L^{(k|n)}$,

(ii) pour tout $\mathbf{x}^1, \ldots, \mathbf{x}^{k+1} \in L^n$:

$$p\left(\bigvee_{i=1}^{k+1} \bigwedge_{j \neq i} \mathbf{x}^j\right) \leq p\big(\mathbf{x}^1\big) \vee \ldots \vee p\big(\mathbf{x}^{k+1}\big),$$

(iii) pour tout $\mathbf{x}^1, \ldots, \mathbf{x}^{k+1} \in \{0,1\}^n$:

$$p\left(\bigvee_{i=1}^{k+1} \bigwedge_{j \neq i} \mathbf{x}^j\right) \leq p\big(\mathbf{x}^1\big) \vee \ldots \vee p\big(\mathbf{x}^{k+1}\big).$$

Démonstration. Soit $p \in \mathcal{P}_L^{(n)}$. Nous allons prouver les implications *(i)* $\Rightarrow$ *(ii)*, *(ii)* $\Rightarrow$ *(iii)* et *(iii)* $\Rightarrow$ *(i)*.

$(i) \Rightarrow (ii)$: Supposons (i). Soit $\mathbf{x}^1, \ldots, \mathbf{x}^{k+1} \in L^n$, et soit L' le plus petit sous-treillis de L qui contient $\mu_p(I)$ pour chaque $I \subseteq [n]$, ainsi que x_i^j pour chaque $i \in [n]$ et chaque $j \in [k+1]$. Nécessairement, L' est un treillis distributif fini (tandis que L peut être infini). En se basant sur le théorème de représentation de Birkhoff, on considérera que L' est un sous-treillis de $\{0,1\}^d$, où d est un entier positif.

Pour chaque $a \in L'$, on dénote la t-ième composante de a by $a|_t$. Sous cette représentation, pour tout $a, b \in L'$, on a

$$a \vee b = (a|_1 \vee b|_1, \ldots, a|_d \vee b|_d), \tag{4.2}$$

$$a \wedge b = (a|_1 \wedge b|_1, \ldots, a|_d \wedge b|_d), \tag{4.3}$$

$$a \leq b \Leftrightarrow \forall t \in [d] : a|_t \leq b|_t. \tag{4.4}$$

Pour tout $\mathbf{x} \in L^n$ on a

$$p(\mathbf{x}) = \bigvee_{I \subseteq [n]} g_I(\mathbf{x}) \quad \text{où} \quad g_I(\mathbf{x}) = \mu_p^*(I) \wedge \bigwedge_{i \in I} x_i,$$

μ_p^* étant la transformée de Möbius ordinale de μ_p. Afin de montrer que (ii) est vérifiée, nous allons montrer que pour tout $I \subseteq [n]$ on a

$$g_I\left(\bigvee_{i=1}^{k+1} \bigwedge_{j \neq i} \mathbf{x}^j\right) \leq g_I(\mathbf{x}^1) \vee \ldots \vee g_I(\mathbf{x}^{k+1}). \tag{4.5}$$

Lorsque $|I| > k$ on a $\mu_p^* = \bot$, et et donc les parties gauches et droites de l'inégalité valent $\bot$. Supposons qu'il existe $I \subseteq [n]$, avec $|I| \leq k$, tel que (4.5) (ii) n'est pas vérifiée. On a alors

$$\mu_p^*(I) \wedge \bigwedge_{z \in I} \left(\bigvee_{i=1}^{k+1} \bigwedge_{j \neq i} x_z^j\right) \not\leq \left(\mu_p^*(I) \wedge \bigwedge_{z \in I} x_z^1\right) \vee \ldots \vee \left(\mu_p^*(I) \wedge \bigwedge_{z \in I} x_z^{k+1}\right),$$

et par conséquent :

$$\bigwedge_{z \in I} \bigvee_{i=1}^{k+1} \bigwedge_{j \neq i} x_z^j \not\leq \left(\bigwedge_{z \in I} x_z^1\right) \vee \ldots \vee \left(\bigwedge_{z \in I} x_z^{k+1}\right).$$

Par (4.4), (4.2), et (4.3), on déduit qu'il existe $t \in [d]$ tel que

$$\bigwedge_{z \in I} \bigvee_{i=1}^{k+1} \bigwedge_{j \neq i} x_z^j|_t \not\leq \left(\bigwedge_{z \in I} x_z^1|_t\right) \vee \ldots \vee \left(\bigwedge_{z \in I} x_z^{k+1}|_t\right).$$

En d'autres termes :

$$\left(\bigwedge_{z \in I} x_z^1|_t\right) \vee \ldots \vee \left(\bigwedge_{z \in I} x_z^{k+1}|_t\right) = 0, \tag{4.6}$$

$$\bigwedge_{z \in I} \bigvee_{i=1}^{k+1} \bigwedge_{j \neq i} x_z^j|_t = 1. \tag{4.7}$$

Il suit de (4.6) que

$$\forall j \in [k+1]\ \exists z \in I, \quad x_z^j|_t = 0.$$

Puisque $k + 1 > |I|$, il existe $z \in I$ tel que

$$\exists j, j' \in [k+1], \quad j \neq j', \quad x_z^j|_t = 0 \quad \text{et} \quad x_z^{j'}|_t = 0.$$

Pour une telle valeur de z on a donc

$$\bigvee_{i=1}^{k+1} \bigwedge_{j \neq i} x_z^j|_t = 0,$$

ce qui contredit (4.7). Par conséquent, (4.5) est vérifié pour tout $I \subseteq [n]$ tel que $|I| \leq k$, ce qui démontre *(i)* $\Rightarrow$ *(ii)*.

$(ii) \Rightarrow (iii)$: Puisque $\{0, 1\} \subseteq L$, toute fonction p qui vérifie *(ii)* vérifie également *(iii)*.

$(iii) \Rightarrow (i)$: On procède par contraposition. Soit une FLP p de degré strictement supérieur à k. Il existe nécessairement $J \subseteq [n]$ tel que $|J| \geq k+1$ et $\mu_p^*(J) > \bot$, autrement dit :

$$\mu_p(J) > \bigvee_{I \subsetneq J} \mu_p(I). \tag{4.8}$$

On dénote par $\mathbf{1}_J$ le n-tuple dont la i-ième composante est $\top$ si $i \in J$ et $\bot$ sinon. On considère $\mathbf{x}^1, \ldots, \mathbf{x}^{k+1}$, où $\mathbf{x}^j = \mathbf{1}_{J \setminus \{j\}}$, pour chaque $i \in [k+1]$. De cette manière on a

$$p\left(\bigvee_{i=1}^{k+1} \bigwedge_{j \neq i} \mathbf{x}^j\right) = p(\mathbf{1}_J) = \mu_p(J) > \bigvee_{I \subsetneq J} \mu_p(I) \geq \bigvee_{j=1}^{k+1} p(\mathbf{1}_{J \setminus \{j\}}) = \bigvee_{j=1}^{k+1} p(\mathbf{x}^j),$$

et la preuve est complète. □

Remarque 55. Ce théorème est une généralisation de la caractérisation des fonctions booléennes croissante de degré au plus k donnée par le Théorème 11.6 de [32].

Remarque 56. Dans le cas où $k = 1$, l'inégalité de l'assertion (ii) devient

$$f\left(\mathbf{x}^1 \vee \mathbf{x}^2\right) \leq f\left(\mathbf{x}^1\right) \vee f\left(\mathbf{x}^2\right),$$

pour tout $\mathbf{x}^1, \mathbf{x}^2 \in L^n$. Puisque p est une FLP (par conséquent non décroissante), cette condition peut être renforcée en une égalité.

Dans le cas où $k = 2$, l'inégalité de l'assertion (ii) devient

$$f\left(\mathrm{med}\left(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3\right)\right) \leq f\left(\mathbf{x}^1\right) \vee f\left(\mathbf{x}^2\right) \vee f\left(\mathbf{x}^3\right),$$

pour tout $\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3 \in L^n$, où $\mathrm{med} : L^3 \to L$ est la *fonction médiane* (appliquée composante à composante) donnée par

$$\mathrm{med}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3).$$

Remarque 57. La condition (iii) est intéressante en particulier lorsque L est un treillis distributif infini, puisque qu'elle permet de vérifier le degré d'une FLP via un nombre fini de tests. Le Théorème 54 généralise la caractérisation des fonctions booléennes non décroissantes de degré au plus k (Theorem 11.6 de [32]).

Remarque 58. On ne peut pas renforcer le Théoreme 54 en remplaçant les inégalités de *(ii)* et *(iii)* par des égalités. Pour le montrer, considérons la FLP p de degré 2 définie par

$$p(x_1, x_2, x_3) = x_1 \vee (x_2 \wedge x_3),$$

ainsi que les tuples $\mathbf{x}^1 = (0,0,0)$, $\mathbf{x}^2 = (0,0,0)$ et $\mathbf{x}^3 = (1,1,1)$. On a

$$p\left(\bigvee_{i=1}^{3} \bigwedge_{j \neq i} \mathbf{x}^j\right) < p\left(\mathbf{x}^1\right) \vee p\left(\mathbf{x}^2\right) \vee p\left(\mathbf{x}^3\right).$$

Remarque 59. Soit p une intégrale de Sugeno k-maxitive. Puisque μ_p est une capacité on a $\mu_p([n]) = 1$ et donc

$$\mu_p([n]) = \bigvee \left\{\mu_p(I) \;\middle|\; I \subseteq [n] \text{ et } |I| \leq k\right\} = 1.$$

Dans le cas où L est un ensemble totalement ordonné, il existe $I \subseteq [n]$ de cardinalité au plus k tel que $\mu(I) = 1$.

4.4 Conclusion

Dans ce Chapitre, nous avons caractérisé les FLP k-maxitives. Cette caractérisation représente une avancée dans l'étude théorique des FLP. Il se peut qu'elle soit utile à la résolution de problèmes futurs, comme par exemple un problème d'interpolation restreint aux FLP k-maxitives.

Deuxième partie

Fonctions d'Utilité de Sugeno pour la classification monotone

Chapitre 5

La classification monotone

La tâche de classification monotone est une restriction de la tâche de classification, qui repose sur des hypothèses similaires à celles de l'approche de la MCDA basée sur les fonctions d'utilité. Cette similarité amène naturellement à envisager l'application à la tâche de classification monotone des fonctions d'agrégation utilisées en MCDA, et notamment des intégrales floues. Ces dernières correspondent à des modèles non additifs permettant l'expression d'interactions entre les attributs. L'auteur de [93] définit des méthodes de classification monotones reposant sur l'intégrale de Choquet et évalue la qualité de leurs prédictions sur des données empiriques. Des études similaires n'ayant jamais été menées concernant l'intégrale de Sugeno, nous comblons partiellement cette lacune dans le Chapitre 6. Le chapitre présent introduit les notions spécifiques de la classification monotone et dresse un état de l'art concis des méthodes existantes pour cette tâche. Nous présentons les Fonctions d'Utilité de Sugeno, qui constituent une classe de fonctions plus générale que les intégrales de Sugeno, et présentons en détail la notion de règle de décision monotone, à laquelle les FUS sont étroitement liées.

5.1 Hypothèses et objectifs

5.1.1 Classification

La classification est une tâche du domaine de l'apprentissage supervisé, dont l'objectif est de déterminer une fonction permettant de classer des observations en fonction de leurs descriptions selon plusieurs attributs.

On considère des ensembles $X_1, \ldots, X_n$ qu'on appelle *domaines des attributs*, ainsi que leur produit Cartésien $\mathbf{X} = X_1 \times \cdots \times X_n$, qu'on appelle *espace des descriptions*. On considère un ensemble fini L dont les éléments sont appelés *classes* ou *labels*. Une *observation* est un couple $(\mathbf{x}, y) \in \mathbf{X} \times L$, dont $\mathbf{x}$ est la description et y est la classe. La description $\mathbf{x}$ est un tuple dont on dénote les valeurs par $(x_1, \ldots, x_n)$. Ces valeurs sont appelées *valeurs d'attributs* de l'observation.

Une fonction $f : \mathbf{X} \to L$ dont le rôle est de prédire la classe d'une observation en fonction de sa description est appelée *classifieur* ou encore *modèle de classification*. Elle est déterminée à partir d'observations dont les classes sont connues, via un algorithme d'apprentissage. Les observations disponibles sont représentées par un ensemble $\mathcal{D} \subseteq \mathbf{X} \times L$ et par une fonction $\eta_{\mathcal{D}} : \mathcal{D} \to \mathbb{N}$ indiquant le nombre d'apparitions de chaque observation (pour tout $(\mathbf{x}, y) \in \mathcal{D}$, on a $\eta_{\mathcal{D}}(\mathbf{x}, y) > 0$).

Pour des raisons de simplicité, on considère ici que les classes sont représentées par des entiers, autrement dit que $L = \{1, \ldots, |L|\}$. Enfin, on dénote par $\mathbf{X}_{\mathcal{D}}$ l'ensemble des descriptions

apparaissant dans les données, autrement dit l'ensemble défini par

$$\mathbf{X}_{\mathcal{D}} = \{\mathbf{x} \mid (\mathbf{x}, y) \in \mathcal{D}\}.$$

Un classifieur f commet une erreur de prédiction sur une observation $(\mathbf{x}, y)$ si $f(\mathbf{x}) \neq y$. Une *fonction de perte* est une fonction $\ell : L^2 \to \mathbb{R}$ qui définit le coût de telles erreurs. Elle associe à chaque $i, j \in L$ le coût d'attribuer une observation à une classe i lorsque cette observation appartient à une classe j (lorsque $i = j$, $\ell(i, j) = 0$). Une fonction de perte largement utilisée par défaut est la fonction ℓ_{0-1} définie par

$$\ell_{0-1}(i, j) = \begin{cases} 0 \text{ si } i = j, \\ 1 \text{ si } i \neq j. \end{cases}$$

L'*erreur empirique* d'un classifieur f associée à ℓ sur un ensemble $\mathcal{D}$ est la perte moyenne de f sur $\mathcal{D}$, définie par

$$\frac{1}{|\eta_{\mathcal{D}}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} \Big[\ell\big(f(\mathbf{x}), y\big)\eta_{\mathcal{D}}(\mathbf{x}, y)\Big],$$

où $|\eta_{\mathcal{D}}|$ dénote le nombre d'observations disponibles, c'est à dire

$$|\eta_{\mathcal{D}}| = \sum_{(\mathbf{x},y)\in\mathcal{D}} \eta_{\mathcal{D}}(\mathbf{x}, y).$$

L'erreur empirique de f associé à ℓ_{0-1} sur $\mathcal{D}$ correspond au taux de mauvaises classifications de f sur $\mathcal{D}$ (*misclassification error rate*, MER) définie par

$$\mathrm{MER}(f, \mathcal{D}) = \frac{1}{|\eta_{\mathcal{D}}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} \Big[\ell_{0-1}\big(f(\mathbf{x}), y\big)\eta_{\mathcal{D}}(\mathbf{x}, y)\Big].$$

Notez qu'une fonction d'erreur peut être utilisée aussi bien durant la phase d'apprentissage, où l'on cherche typiquement à apprendre un classifieur en minimisant son erreur empirique sur les données d'apprentissage, que pour l'évaluation de la fiabilité du modèle, qui est réalisée sur des données non utilisées pendant l'apprentissage.

5.1.2 Classification monotone

La *classification monotone*, aussi appelée *classification isotone* ou *classification ordinale avec contraintes de monotonies* est un cas particulier de la tâche de classification, où

- les ensembles $X_1, \ldots, X_n$ et L sont totalement ordonnés ;
- on restreint l'ensemble des classifieurs acceptables aux fonctions non-décroissantes (*contrainte de monotonie*).

La contrainte de monotonie est justifiée par une connaissance à priori sur les données, qui permet d'affirmer que chaque attribut est corrélé positivement à la valeur de la classe. Cette connaissance a priori justifie l'utilisation d'algorithmes d'apprentissage différents de ceux employés pour la tâche de classification standard. Certains de ces algorithmes et des formalismes qui les sous-tendent seront traités dans la suite de ce chapitre.

Certaines approches supposent que $X_1, \ldots, X_n$ sont des intervalles numériques dont les valeurs représentent des quantités. Nous considérons ici un cadre général où ces ensembles peuvent être numériques ou non. En revanche, nous supposons que les ensembles $X_1, \ldots, X_n$ et L sont bornés. Les bornes inférieures de ces ensembles seront dénotées par un même symbole 0, tandis que leurs bornes supérieures seront dénotées par 1.

Remarque 60. Le terme *classification monotone* ne devrait pas être confondu avec *classification ordinale*, qui désigne une tâche plus générale que la classification monotone, dans laquelle l'ensemble des classes est totalement ordonné, mais où les domaines des attributs ne le sont pas nécessairement et où le modèle recherché n'est pas nécessairement une fonction monotone (voir par exemple [60]).

Le fait que l'ensemble des classes soit totalement ordonné justifie l'utilisation de fonctions de perte spécifiques pour l'évaluation des classifieurs. Bien qu'il soit possible d'évaluer un classifieur monotone d'après la fonction de perte ℓ_{0-1}, il semble pertinent d'imposer, pour toutes classes $i < j < k$, que la fonction de perte utilisée respecte $\ell(i,j) < \ell(i,k)$. Les fonctions de perte qui satisfont l'inégalité $\ell(i,j) \leq \ell(i,k)$ pour toutes classes $i < j < k$ sont appelées *fonctions de perte en V* (*V-shaped loss function* [72]). Lorsqu'on ne dispose pas d'informations permettant de déterminer une distance spécifique entre chaque paire de classes, il est commun de choisir par défaut la fonction de perte ℓ_1 définie par

$$\ell_1(i,j) = |i-j|.$$

L'erreur empirique associée à cette fonction de perte est équivalent à l'*erreur absolue moyenne* (*mean absolute error*, MAE) lorsque les classes de L sont traitées comme des valeurs quantitatives. La MAE de f sur $\mathcal{D}$ est définie par

$$\mathrm{MAE}(f,\mathcal{D}) = \frac{1}{|\eta_{\mathcal{D}}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} \Big[\ell_1(f(\mathbf{x}),y)\eta_{\mathcal{D}}(\mathbf{x},y)\Big].$$

La MER et la MAE étant largement utilisées dans le domaine de la classification ordinale, nous évaluerons nos propres modèles selon ces deux mesures.

Remarque 61. La MER et la MAE mesurent la précision absolue du classifieur. Des mesures de l'erreur comme le C-index [48] ou le τ de Kendall (voir par exemple [1], Chapitre 7) mesurent la similarité entre l'ordre des observations prédit par le classifieur (une observation est supérieure à une autre si sa classe est supérieure), avec l'ordre réel des observations. De telles mesures sont plus adaptées à l'évaluation de modèles de ranking et ne seront pas utilisées ici. Notez enfin que certaines mesures [20] visent à rendre compte simultanément de la précision absolue des prédictions et de la fidélité de l'ordre prédit par le classifieur.

5.1.3 L'interprétabilité des modèles

L'objectif premier de la tâche de classification est d'obtenir un classifieur fiable qui commette peu d'erreurs. Cependant, dans certains cadres d'application, il est utile que le modèle puisse être compris par un utilisateur humain : on dit alors que le modèle est *interprétable*. La question de l'interprétabilité des modèles est une large problématique du domaine de l'apprentissage automatique (voir par exemple [68]). Dans le cadre où nous nous plaçons, avoir connaissance de la monotonie du modèle constitue déjà une information importante qui assure un certain degré d'interprétabilité, quel que soit le type de modèle monotone utilisé. Cependant, cette connaissance ne suffit pas toujours à expliquer, par exemple, pourquoi une observation donnée est affectée par le modèle à une classe en particulier. Pour cela, on a besoin d'informations supplémentaires sur le modèle ; par exemple d'une quantification de l'importance accordée à chaque attribut.

Dans la suite, nous distinguerons deux sens possibles du terme *interprétabilité* :

- D'une part, la possibilité, pour un utilisateur, de comprendre la raison d'une prédiction donnée ; nous appelons cette propriété l'*interprétabilité ponctuelle* du modèle.

- D'autre part, le degré auquel le modèle lui-même aide à la compréhension générale du phénomène qu'il modélise ; par exemple, un modèle basé sur la moyenne pondérée fournit une information sur l'importance relative des attributs à travers les poids qu'il leur accorde à chacun. Nous appelons cette propriété *interprétabilité globale*.

5.2 Monotonie et anti-monotonie dans les données

5.2.1 Données monotones

On dit qu'un ensemble d'observations $\mathcal{D}$ est monotone si pour toute paire $\{(\mathbf{x}, y), (\mathbf{x'}, y')\}$ dans $\mathcal{D}$ on a

$$\mathbf{x} \leq \mathbf{x'} \quad \Longrightarrow \quad y \leq y'.$$

Dans ce cas les fonctions $\lambda_{\mathcal{D}}^{-} : \mathbf{X} \to L$ et $\lambda_{\mathcal{D}}^{+} : \mathbf{X} \to L$ définies par

$$\lambda_{\mathcal{D}}^{-}(\mathbf{x}) = \bigvee \left\{ y' \;\middle|\; (\mathbf{x'}, y') \in \mathcal{D} \text{ et } \mathbf{x'} \leq \mathbf{x} \right\} \quad \text{et} \quad \lambda_{\mathcal{D}}^{+}(\mathbf{x}) = \bigwedge \left\{ y' \;\middle|\; (\mathbf{x'}, y') \in \mathcal{D} \text{ et } \mathbf{x} \leq \mathbf{x'} \right\}$$

sont (respectivement) le plus petit et le plus grand classifieur qui ne font aucune erreur sur $\mathcal{D}$. Pour tout $\mathbf{x} \in \mathbf{X}$ on a $\lambda_{\mathcal{D}}^{-}(\mathbf{x}) \leq \lambda_{\mathcal{D}}^{+}(\mathbf{x})$, et pour tout $(\mathbf{x}, y) \in \mathcal{D}$ on a $\lambda_{\mathcal{D}}^{-}(\mathbf{x}) = \lambda_{\mathcal{D}}^{+}(\mathbf{x}) = y$. On dit qu'une fonction f interpole $\mathcal{D}$ si $f(\mathbf{x}) = y$ pour tout $(\mathbf{x}, y) \in \mathcal{D}$. L'intervalle $[\lambda_{\mathcal{D}}^{-}(\mathbf{x}), \lambda_{\mathcal{D}}^{+}(\mathbf{x})]$ contient l'ensemble des fonctions non-décroissantes qui interpolent $\mathcal{D}$. Lorsque les données sont monotones, la plupart des algorithmes d'apprentissage de classifieurs monotones cités dans la section suivante retournent une fonction de cet intervalle.

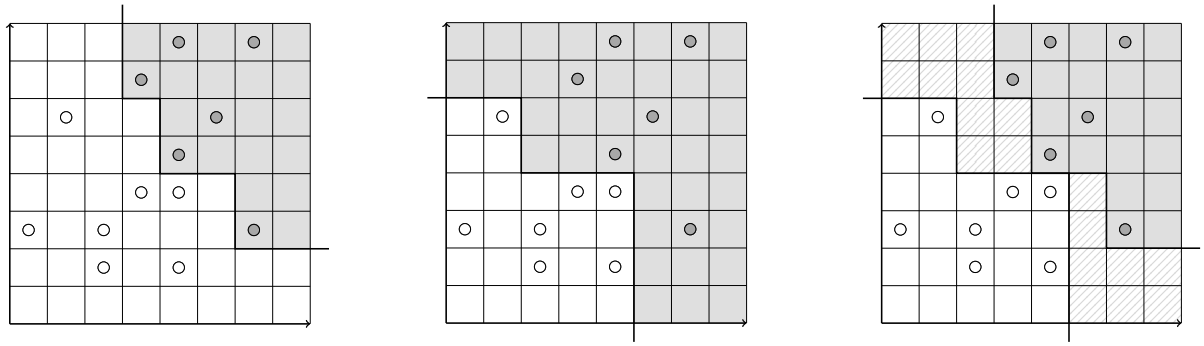


FIGURE 5.1 – Exemple de données d'apprentissage monotones ; deux attributs aux domaines discrets, deux classes (0 et 1). Chaque description possible est représentée par une case de la grille. Les observations de la classe 0 (resp. 1) sont représentées par des points blancs (resp. gris). Le premier graphique représente la fonction $\lambda_{\mathcal{D}}^{-}$ par une frontière séparant les descriptions que $\lambda_{\mathcal{D}}^{-}$ associe à 0 (dans la zone blanche) de celles qu'elle associe à 1 (dans la zone grise). Le second graphique représente $\lambda_{\mathcal{D}}^{+}$ de manière analogue. Le troisième graphique représente l'intervalle $[\lambda_{\mathcal{D}}^{-}, \lambda_{\mathcal{D}}^{+}]$: les zones blanches et grises correspondent respectivement aux descriptions qui sont associées à 0 et 1 par toutes les fonctions de l'intervalle. La zone hachurée contient les descriptions $\mathbf{x} \in \mathbf{X}$ pour lesquelles $\lambda_{\mathcal{D}}^{-}(\mathbf{x}) < \lambda_{\mathcal{D}}^{+}(\mathbf{x})$.

5.2.2 Anti-monotonie

On dit qu'une paire d'observations $\{(\mathbf{x}, y), (\mathbf{x'}, y')\}$ est anti-monotone si

$$\mathbf{x} \leq \mathbf{x'} \quad \text{et} \quad y' < y.$$

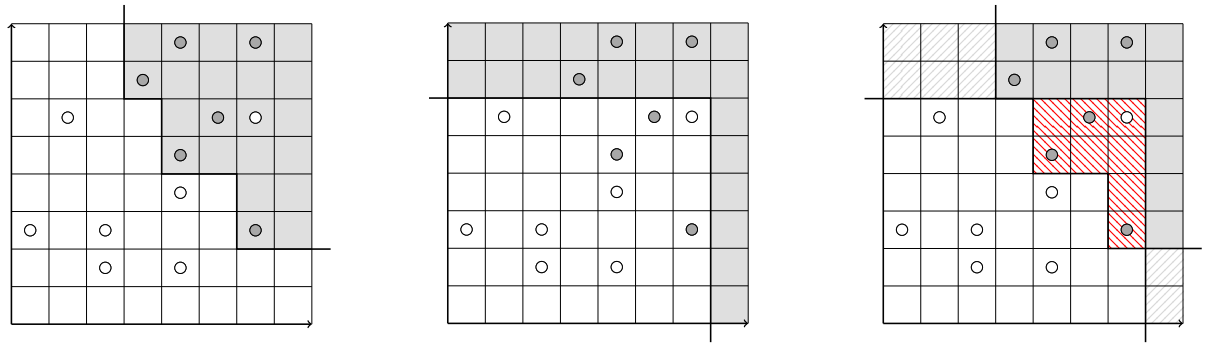


FIGURE 5.2 – Exemple de données d'apprentissage non monotones. Les deux premiers graphiques représentent $\lambda^-_{\mathcal{D}}$ et $\lambda^+_{\mathcal{D}}$. L'intervalle $[\lambda^-_{\mathcal{D}}, \lambda^+_{\mathcal{D}}]$ est vide. Pour le troisième graphique : dans la zone blanche on a $\lambda^-_{\mathcal{D}}(\mathbf{x}) = \lambda^+_{\mathcal{D}}(\mathbf{x}) = 0$; dans les zones grises on a $\lambda^-_{\mathcal{D}}(\mathbf{x}) = \lambda^+_{\mathcal{D}}(\mathbf{x}) = 1$, dans les zones hachurées en gris $\lambda^-_{\mathcal{D}}(\mathbf{x}) < \lambda^+_{\mathcal{D}}(\mathbf{x})$, et dans les zones hachurées en rouge $\lambda^+_{\mathcal{D}}(\mathbf{x}) < \lambda^-_{\mathcal{D}}(\mathbf{x})$.

Remarquez que pour une telle paire on a

$$\lambda^+_{\mathcal{D}}(\mathbf{x}) < \lambda^-_{\mathcal{D}}(\mathbf{x}) \quad \text{et} \quad \lambda^+_{\mathcal{D}}(\mathbf{x'}) < \lambda^-_{\mathcal{D}}(\mathbf{x'}).$$

Il n'existe alors aucune fonction monotone qui interpole $\mathcal{D}$. L'existence de paires d'observations anti-monotones, en supposant qu'elle ne remette pas en cause la pertinence d'un classifieur monotone, peut être interprétée de plusieurs manières : soit les informations données par les valeurs d'attributs de chaque observation sont insuffisantes (il existe d'autres paramètres qui ne sont pas représentés dans les données), soit les données contiennent des erreurs. Dans les deux cas, la présence de paires anti-monotones peut compliquer la tâche d'apprentissage du classifieur. Ce problème peut être surmonté de plusieurs manières.

- Réaliser l'apprentissage d'un classifieur monotone sur les données non monotones, en cherchant à minimiser l'erreur empirique du classifieur sur les données d'apprentissage.
- Pré-traiter les données afin de les rendre monotones (en effectuant des changements minimaux).
- Formaliser l'incertitude qui découle de l'anti-monotonie des données, afin de l'intégrer au modèle.

La première approche est une approche classique en machine learning, qui consiste à rechercher un classifieur dont l'erreur empirique est minimale sur les données d'apprentissage, tout en imposant certaines contraintes (ici, la monotonie). Nous présentons donc les deux autres approches dans les sous-sections qui suivent.

5.2.3 Restauration de la monotonie

Soit $\mathcal{D}$ un ensemble d'apprentissage non monotone. On souhaite rendre $\mathcal{D}$ monotone en effectuant un minimum de transformations.

Une approche possible est de retirer certaines observations de $\mathcal{D}$, de manière à ce qu'aucune paire anti-monotone ne subsiste. Cela revient à rechercher un sous ensemble monotone de $\mathcal{D}$. On cherche dans ce cas à minimiser le nombre d'observations supprimées, c'est à dire à maximiser la taille du sous-ensemble monotone retourné.

Problème de sous-ensemble monotone maximal. *Pour un ensemble d'apprentissage $\mathcal{D}$ donné, retourner un sous-ensemble monotone de $\mathcal{D}$ de taille maximale.*

Ce problème peut aisément être réduit à celui de trouver un *sous-ensemble indépendant maximal* (maximum independant set) dans un graphe, c'est à dire trouver l'un des plus grands ensembles de nœuds n'ayant aucun arc en commun. On considère le graphe orienté $G = (\mathcal{D}, A)$ dont l'ensemble des noeuds est $\mathcal{D}$ et dont l'ensemble des arcs est

$$A = \Big\{ \big((\mathbf{x}, y), (\mathbf{x'}, y')\big) \;\Big|\; \mathbf{x} \leq \mathbf{x'} \text{ et } y' < y \Big\}.$$

Bien que le problème de sous ensemble indépendant maximal soit NP-complet [92] dans le cas général, il peut ici être résolu en $O(m^3)$ [85]. Cela est dû au fait que A est une relation transitive ; le problème de sous-ensemble indépendant maximal peut alors être reformulé en un problème de flot maximal (voir [66] ou encore [80], Section 1.5).

Une méthode alternative et qui fait l'objet d'un plus grand nombre de travaux est la relabélisation. Elle consiste à modifier la classe de certaines observations, de manière à restaurer la monotonie des données. Plus précisément, on cherche une fonction non décroissante $\pounds : D \to L$ appelée *relabeling*, déterminant un nouveau label pour chaque observation, en fonction de ses valeurs d'attributs. Notez qu'il est raisonnable d'imposer la contrainte suivante à tout relabeling $\pounds$:

$$\forall (\mathbf{x}, y) \in \mathcal{D}, \quad \lambda^+_{\mathcal{D}}(\mathbf{x}) \leq \pounds(\mathbf{x}) \leq \lambda^-_{\mathcal{D}}(\mathbf{x}) \tag{5.1}$$

En effet, $\lambda^+_{\mathcal{D}}(\mathbf{x})$ étant le plus petit label associé (dans $\mathcal{D}$) à une description au moins égale à $\mathbf{x}$, aucune information dans $\mathcal{D}$ ne justifie de ré-attribuer une valeur plus faible que $\lambda^+_{\mathcal{D}}(\mathbf{x})$ à $\mathbf{x}$. La contrainte $\pounds(\mathbf{x}) \leq \lambda^-_{\mathcal{D}}(\mathbf{x})$ se justifie de manière duale.

Parmi les fonctions qui satisfont la contrainte (5.1) se trouvent les relabelings qui minimisent l'erreur empirique associée à une de fonction de perte en V.

Problème de relabeling optimal. *Soit un ensemble non-monotone $\mathcal{D}$ et une fonction de perte ℓ. Déterminer un relabeling $\pounds : \boldsymbol{X} \to L$ dont l'erreur empirique associée à ℓ sur $\mathcal{D}$ est minimal.*

Un relabeling optimal peut être obtenu en $O(m^3|L|)$ pour toutes les fonctions de perte convexes (parmi lesquelles la fonction de perte ℓ_1, dont l'erreur empirique associée correspond à la MAE), via l'algorithme présenté dans [44]. Un relabeling optimal peut être obtenu en $O(m^3|L|^3)$ pour toutes les fonctions de perte en V [85].

Plusieurs études [34, 35, 71], effectuées sur des données empiriques et sur des données artificielles volontairement bruitées [45], montrent qu'effectuer un relabeling sur les données d'apprentissage augmente, dans certains cas, la précision des modèles entrainés sur ces données.

Notez enfin que les données obtenues après un relabeling optimal peuvent être interpolées par une fonction non-décroissante. Cette méthode d'apprentissage en deux étapes (relabeling et interpolation) permet d'obtenir un classifieur qui minimise l'erreur empirique sur les données d'apprentissage.

Dans la suite, on dénotera par $\mathcal{M}$ les ensembles de données qui ont subit un relabeling.

5.2.4 Prise en compte de l'incertitude

La Dominance-based Rough Set Approach (DRSA) [57] est un cadre théorique basé sur la théorie des rough sets [81], pensé pour l'analyse de problèmes de classification monotone (en particulier dans le cadre de la MCDA). Tandis que la notion de relabeling est plutôt fondée sur l'idée que les données sont bruitées, la DRSA étend le cadre théorique des rough sets, où certains objets sont considérés indiscernables par manque d'information. Les paires anti-monotones, dans ce cadre, sont vues comme la conséquence d'une incertitude qu'il est utile de conserver et de

représenter dans le modèle appris à partir des données. Nous présentons maintenant les notions de bases de la DRSA (en adaptant la notation standard, afin de faciliter la lecture).

La DRSA considère une relation d'ordre $\preccurlyeq$ sur l'ensemble des observations, appelée *relation de dominance*, définie par

$$(\mathbf{x}, y) \preccurlyeq (\mathbf{x'}, y') \quad \text{si} \quad \mathbf{x} \leq \mathbf{x'}.$$

Pour chaque $o \in \mathcal{D}$, on définit les ensembles

$$D^-(o) = \{o' \in \mathcal{D} \mid o' \preccurlyeq o\} \quad \text{et} \quad D^+(o) = \{o' \in \mathcal{D} \mid o \preccurlyeq o'\}.$$

Soit $k \in L$. On appelle C_k l'ensemble des observations dont la classe est k. Afin de prendre en compte la nature ordinale des classes de L, on considère les ensembles $C_1^{\geq}, \ldots, C_{|L|}^{\geq}$ définis par

$$C_k^{\geq} = \bigcup \{C_i \mid i \in L,\ i \geq k\}$$

pour tout $k \in [n]$. L'ensemble $C_k^{\geq}$ (où $k \in L$) est l'ensemble des objets de classe au moins k. Le principe central de la DRSA est de considérer deux approximations de $C_k^{\geq}$ appelées approximation basse et approximation haute, respectivement définies par

$$\underline{C_k^{\geq}} = \left\{o \in D \;\middle|\; D^+(o) \subseteq C_k^{\geq}\right\} \quad \text{et} \quad \overline{C_k^{\geq}} = \left\{o \in D \;\middle|\; D^-(o) \cap C_k^{\geq} \neq \emptyset\right\}$$

Ces approximations de $C_k^{\geq}$ permettent de gérer les paires anti-monotones de la manière qui suit. Soit $(\mathbf{x}, y)$ et $(\mathbf{x'}, y')$ deux observations telles que $\mathbf{x} \leq \mathbf{x'}$ et $y' < y$. On a

$$(\mathbf{x}, y), (\mathbf{x'}, y') \in \overline{C_y^{\geq}} \quad \text{et} \quad (\mathbf{x}, y), (\mathbf{x'}, y') \notin \underline{C_y^{\geq}}$$

Dans le cas où les données sont monotones, on a $\underline{C_k^{\geq}} = \overline{C_k^{\geq}}$. En fait, pour tout $(\mathbf{x}, y) \in \mathcal{D}$ et $k \in L$ on a

$$\left[(\mathbf{x}, y) \in \underline{C_k^{\geq}} \iff \lambda_{\mathcal{D}}^+(\mathbf{x}) \geq k\right] \quad \text{et} \quad \left[(\mathbf{x}, y) \in \overline{C_k^{\geq}} \iff \lambda_{\mathcal{D}}^-(\mathbf{x}) \geq k\right].$$

L'approximation basse de $C_k^{\geq}$ peut donc être vue comme l'ensemble des observations auxquelles tout relabeling acceptable associe une classe au moins égale à k. L'approximation haute de $C_k^{\geq}$ peut être vue comme l'ensemble des observation auquel au moins un relabeling acceptable associe une classe au moins égale à k.

L'approche classique en DRSA consiste à apprendre un modèle de classification à partir des approximations basses et un modèle à partir des approximations hautes. Ces deux modèles donnent respectivement une prédiction basse et une prédiction haute de la classe de chaque observation. Un défaut de cette approche est sa grande sensibilité aux outliers ; une seule observation mal labelisée peut en effet rendre vide certaines approximations basses (considérez, par exemple, le cas extrême où les données contiennent une observation dont la description est $(0, \ldots, 0)$ et la classe est 1).

Afin de corriger ce défaut, différentes approches probabilistes des rough sets ont été développées. Celles-ci construisent des approximations basses et hautes des ensembles dont les définitions sont moins strictes, et donc plus résistantes au bruit. La Variable Consistency Dominance-based Rough Set Approaches (VC-DRSA) [18] est l'une de ces approches, dans laquelle l'estimation de basse de $C_k^{\geq}$ est donnée par une expression de la forme

$$\underline{C_k^{\geq}} = \{o \in \mathcal{D} \mid \Theta(o, k) \propto \theta_k\} \tag{5.2}$$

où $\propto$ est la relation $\leq$ (resp. $\geq$), Θ est une *mesure de consistence* qui associe un degré de confiance au fait que o appartient (resp. n'appartient pas) à la classe k, et θ_k est un seuil.

La VC-DRSA sert notamment de base à l'algorithme d'apprentissage de règles de décision monotones VC-DomLEM, que nous présentons brièvement dans la Sous-section 5.3.4.

5.3 Modèles et algorithmes

5.3.1 Adaptations de modèles standards

Certains algorithmes de classification monotone sont obtenus par l'adaptation d'un algorithme de classification standard. Afin de garantir que le classifieur obtenu est une fonction non décroissante, on ajoute des contraintes qui restreignent l'ensemble des modèles acceptables. Cette méthode a été appliquée avec plusieurs algorithmes classiques, tels que les k-plus proches voisins [43], les réseaux bayésiens [2, 46, 62] ou les réseaux de neurones ([90] où une architecture spécifique est proposée, et [103]).

Dans cette section nous présentons certains algorithmes de classification monotone existants, en privilégiant ceux qui produisent des modèles interprétables ou qui sont pensés spécifiquement pour traiter les problèmes de classification monotone.

Remarque 62. Les algorithmes d'apprentissage de classifieurs monotones sont à distinguer des algorithmes d'apprentissage de classifieurs quasi-monotones, où la monotonie du classifieur est vue comme une propriété à maximiser, et non comme une contrainte stricte (voir par exemple [22, 39] dans le cas des SVM ou [50] dans le cas des forêts aléatoires).

Arbres de décision binaire

La plupart des algorithmes d'apprentissage d'arbres de décision non monotones, y compris les algorithmes standards CART [15] et C4.5 [84], fonctionnent selon un schéma similaire : l'ensemble des descriptions est partitionné récursivement, jusqu'à ce que les sous-ensembles de $\mathbf{X}$ ainsi créés soient suffisamment homogènes en terme de classes. Chacun de ces sous-ensembles correspond alors à une feuille de l'arbre, tandis que chacun des nœuds internes correspond à l'une des conditions définissant le partitionnement récursif de $\mathbf{X}$. Ce processus est guidé par

- une règle d'arrêt, qui détermine si un sous-ensemble donné doit être partitionné ou devenir une feuille de l'arbre,
- une règle de division, qui détermine une partition d'un ensemble donné,
- une règle de labeling, qui attribue une classe à chaque feuille de l'arbre.

Lorsque les attributs sont des ensembles ordonnés, comme c'est le cas dans la classification monotone, les conditions correspondant aux nœuds internes sont de la forme $x_i \geq \alpha_i$, où $\alpha_i \in X_i$.

Il est possible de définir des règles d'arrêt, de division et de labeling qui garantissent l'apprentissage d'un arbre monotone. C'est le cas de la méthode décrite dans [82], qui retourne un arbre monotone décrivant une fonction de l'intervalle $[\lambda^-_{\mathcal{D}}, \lambda^+_{\mathcal{D}}]$ (cette méthode suppose que les données $\mathcal{D}$ sont monotones). Un autre exemple est l'algorithme REMT [63], qui réalise l'apprentissage d'un arbre de décision sur la base d'une mesure appelée *rank entropy*. Cette mesure généralise l'entropie de Shannon de manière à prendre en compte l'ordre des valeurs d'attributs et des classes. Cet algorithme garantit l'apprentissage d'un arbre monotone, à condition que les données d'apprentissage soient monotones. Ces deux approches, lorsqu'elles sont précédées par un relabeling des données d'apprentissage, permettent d'apprendre des arbres monotones, que les données soient monotones ou non.

Il est également possible de réaliser l'apprentissage d'un arbre de décision standard, qui est ensuite « monotonisé ». Le principe de l'algorithme ICT [97] est de construire un arbre en utilisant une variante de CART, puis de relabéliser les feuilles de manière à rendre l'arbre monotone.

De par leur nature, les arbres de décision binaires offrent une bonne interprétabilité ponctuelle : chaque prédiction de l'arbre est justifiée par une branche unique, qui correspond à une série de conditions simples (typiquement de la forme $x_i \geq \alpha_i$, dans le cas des arbres de décision monotones). L'interprétabilité globale du modèle dépend grandement du nombre de nœuds présents dans l'arbre.

Notez cependant que de faibles changements dans les données d'apprentissage peuvent avoir un fort impact sur la forme de l'arbre, rendant l'interprétation de l'arbre potentiellement trompeuse (voir [61], Section 9.2).

5.3.2 La séparation isotone

Une partie des algorithmes de classification monotone sélectionnent, de manière implicite ou explicite, leur classifieur dans l'intervalle $[\lambda^-_{\mathcal{D}}, \lambda^+_{\mathcal{D}}]$ (après avoir restauré la monotonie des données si nécessaire). Un exemple d'une telle approche est la séparation isotone [21], qui est une méthode de classification monotone binaire. La *séparation* des deux classes (dénotées par 0 et 1) est effectuée de la manière suivante. Les données sont relabelisées en un ensemble $\mathcal{M}$. Pour toutes les descriptions $\mathbf{x} \in \mathbf{X}$ où $\lambda^-_{\mathcal{M}}(\mathbf{x})$ et $\lambda^+_{\mathcal{M}}(\mathbf{x})$ donnent la même classe y (zones blanches et grises dans la Figure 5.1), le classifieur retourne y. Pour toutes les autres observations $\mathbf{x} \in \mathbf{X}$ (zone hachurée dans la Figure 5.1), le choix de la classe est fait d'après une mesure de distance entre les descriptions : si $\mathbf{x}$ est « plus proche » des descriptions de $\{\mathbf{x'} \in \mathbf{X} \mid \lambda^+_{\mathcal{M}}(\mathbf{x'}) = 0\}$ que des descriptions de $\{\mathbf{x'} \in \mathbf{X} \mid \lambda^-_{\mathcal{M}}(\mathbf{x'}) = 1\}$, alors le classifieur renvoie 0 ; sinon le classifieur renvoie 1.

La séparation isotone peut ensuite être généralisée aux tâches de classification monotone à plus de deux classes, via la combinaison de $|L| - 1$ classifieurs binaires.

La séparation isotone possède plusieurs avantages, notamment celui de distinguer les descriptions dont la classe est vue comme certaine (lorsque $\lambda^-_{\mathcal{M}}(\mathbf{x}) = \lambda^+_{\mathcal{M}}(\mathbf{x})$) et les autres. De plus, la séparation isotone montre généralement une bonne précision sur les données de tests [70]. Cependant, la séparation des classes repose sur une notion de distance, ce qui ne permet pas de fournir une justification compréhensible des prédictions du modèles pour les descriptions sur lesquelles $\lambda^-_{\mathcal{M}}$ et $\lambda^+_{\mathcal{M}}$ diffèrent.

5.3.3 OSDL et MOCA

OSDL [73] et MOCA [7, 6] sont deux approches probabilistes de la classification monotone fondés sur des principes similaires. Nous faisons ici une description succincte de MOCA. Les deux approches font appel à une fonction $\hat{F} : \mathbf{X} \times L \to [0, 1]$, appelée *fonction de distribution de probabilité cumulée*, définie par

$$\hat{F}(\mathbf{x}, y) = \frac{\sum_{y' \leq y} \eta_{\mathcal{D}}(\mathbf{x}, y')}{\eta_{\mathcal{D}}(\mathbf{x})}.$$

où $\eta_{\mathcal{D}}(\mathbf{x})$ est le nombre d'observations dont la description est $\mathbf{x}$, et $\eta_{\mathcal{D}}(\mathbf{x}, y')$ est le nombre d'observations dont la description est $\mathbf{x}$ et la classe est y' (voir Sous-section 5.1.1). Lorsque les données sont monotones on a l'implication suivante :

$$\mathbf{x} \leq \mathbf{x'} \quad \Longrightarrow \quad \hat{F}(\mathbf{x'}, y) \leq \hat{F}(\mathbf{x}, y)$$

pour tout $\mathbf{x}, \mathbf{x'} \in \mathbf{X}_{\mathcal{D}}$ et $y \in L$ (la distribution de probabilité cumulée est non croissante sur $\mathbf{X}_{\mathcal{D}}$). À partir de $\hat{F}$, la fonction monotone F^* est définie par

$$F^*(\mathbf{x}, y) = \begin{cases} g^*(\mathbf{x}, y) \text{ si } y \in [|L| - 1], \\ 1 \text{ sinon,} \end{cases}$$

où $g^* : \mathbf{X} \times L \to [0, 1]$ est la fonction non croissante sur $\mathbf{X}$ qui minimise la valeur de l'expression suivante :

$$\sum_{(\mathbf{x}, y) \in \mathcal{D}} \left| g^*(\mathbf{x}, y) - \hat{F}(\mathbf{x}, y) \right|^2 \eta_{\mathcal{D}}(\mathbf{x}, y).$$

Cette étape peut être vue comme l'équivalent d'un relabeling effectué sur les distributions de probabilité, au lieu des classes des observations. Cette distribution cumulée est ensuite étendue par interpolation aux descriptions non observées. La plus petite et la plus grande interpolations monotones de F^* possibles sont les fonctions F^- et F^+ définies par

$$F^-(\mathbf{x}, y) = \bigvee \left\{ F^*(\mathbf{x'}, y) \,\middle|\, \mathbf{x'} \in D \text{ et } \mathbf{x} \leq \mathbf{x'} \right\} \quad \text{et} \quad F^+(\mathbf{x}, y) = \bigwedge \left\{ F^*(\mathbf{x'}, y) \,\middle|\, \mathbf{x'} \in D \text{ et } \mathbf{x'} \leq \mathbf{x} \right\}.$$

Remarquez que les fonctions F^- et F^+ jouent ici un rôle similaire à $\lambda^-_{\mathcal{M}}$ et $\lambda^+_{\mathcal{M}}$ (cependant, puisque F^* est décroissante on a $F^+(\mathbf{x}, y) \leq F^-(\mathbf{x}, y)$ pour tout $\mathbf{x} \in \mathbf{X}$ et $y \in L$). De plus, pour tout $\mathbf{x} \in \mathbf{X}_{\mathcal{D}}$ et $y \in L$, on a

$$F^+(\mathbf{x}, y) = F^-(\mathbf{x}, y).$$

Pour un élément $\mathbf{x} \in \mathbf{X} \backslash \mathbf{X}_{\mathcal{D}}$ on peut avoir $F^+(\mathbf{x}, y) < F^-(\mathbf{x}, y)$. Une distribution de probabilité cumulée unique est associée à chaque élément de $\mathbf{x} \in \mathbf{X}$ par l'interpolation $F : \mathbf{X} \times L \to [0, 1]$ de la fonction F^* définie par

$$F(\mathbf{x}, y) = \alpha F^-(\mathbf{x}, y) + (1 - \alpha) F^+(\mathbf{x}, y),$$

où α est un paramètre appartenant à $[0, 1]$. Dans cette méthode, la distribution de probabilité associée à chaque description $\mathbf{x} \in \mathbf{X}$ est une moyenne pondérée de la plus petite et de la plus grande distributions possibles pour $\mathbf{x}$ (données respectivement par F^+ et F^-).

5.3.4 Ensembles de règles de décision

La séparation isotone, MOCA et OSDL ont pour désavantage commun d'être peu interprétables, car faisant appel à des notions de distance ou de probabilité appliquées directement aux observations présentes dans les données d'apprentissage.

Les ensembles de règles de décision, au contraire, constituent des modèles clairement distincts des données à partir desquelles ils sont appris. De plus, chaque prédiction d'un ensemble de règles peut être justifiée par une ou plusieurs règles ; ce type de modèle est souvent considéré comme interprétable, à condition de satisfaire certaines conditions (par exemple, d'être constitué de règles dont les conditions sont de taille raisonnable).

Dans le chapitre suivant, les ensembles de règles de décision serviront de base à l'étude de modèles basés sur l'intégrale de Sugeno. Par conséquent nous présentons ce formalisme des ensembles de règles de décision de manière détaillée.

Définitions

Nous appelons *règles de décision monotones* (ou simplement *règles de décision*) deux types de règles : les règles de sélection et les règles de rejet. Soit $\delta \in L$ et, pour chaque $i \in [n]$, $\alpha_i \in X_i$. Une *règle de sélection* de $\mathbf{X}$ vers L est une implication de la forme

$$x_1 \geq \alpha_1,\ \ldots,\ x_n \geq \alpha_n \quad \Longrightarrow \quad y \geq \delta. \tag{5.3}$$

Une *règle de rejet* de $\mathbf{X}$ vers L est une implication de la forme

$$x_1 \leq \alpha_1,\ \ldots,\ x_n \leq \alpha_n \quad \Longrightarrow \quad y \leq \delta. \tag{5.4}$$

Les règles de la forme (5.3) et (5.4) seront parfois exprimées sous une forme abrégée par, respectivement,

$$(\alpha_1, \ldots, \alpha_n) \nearrow \delta \quad \text{et} \quad (\alpha_1, \ldots, \alpha_n) \searrow \delta.$$

De plus, pour toute règle de décision r, on dénotera par $\alpha_1^r, \ldots, \alpha_n^r$ et δ^r les valeurs telles que $r = \left[(\alpha_1^r, \ldots, \alpha_n^r) \nearrow \delta^r\right]$ (si r est une règle de sélection) ou $r = \left[(\alpha_1^r, \ldots, \alpha_n^r) \searrow \delta^r\right]$ (si r est une règle de rejet). Notez que certains seuils α_i^r peuvent correspondre à des conditions qui sont toujours vérifiées.

Dans une règle de sélection, le seuil $\alpha_i^r = 0$ correspond à la condition $x_i \geq 0$. Un attribut $i \in [n]$ est *actif* si $\alpha_i > 0$. L'ensemble des attributs actifs d'une règle de sélection r est dénoté A^r et défini par

$$A^r = \{i \in [n] \mid \alpha_i^r > 0\}.$$

La règle r peut être exprimée d'après ses attributs actifs sous la forme

$$\forall i \in A^r,\ x_i \geq \alpha_i^r \quad \Longrightarrow \quad y \geq \delta^r$$

Dans une règle de rejet, le seuil $\alpha_i^r = 1$ correspond à la condition $x_i \leq 1$. Un attribut $i \in [n]$ est *actif* si $\alpha_i < 1$. L'ensemble des attributs actifs d'une règle de rejet r est dénoté A^r et défini par

$$A^r = \{i \in [n] \mid \alpha_i^r < 1\}.$$

La règle r peut être exprimée d'après ses attributs actifs sous la forme

$$\forall i \in A^r,\ x_i \leq \alpha_i^r \quad \Longrightarrow \quad y \leq \delta^r$$

Dans la suite de ce document, nous ne considérerons que des ensembles dont les règles sont toutes du même type (sélection ou rejet). Les ensembles de règles de sélection et les ensembles de règles de rejet seront respectivement appelés *select-sets* et *reject-sets*. Ces ensembles définissent des fonctions de $\mathbf{X}$ à L.

Pour toute règle de sélection r, on définit la fonction $f_r : \mathbf{X} \to L$ par

$$f_r(x_1, \ldots, x_n) = \begin{cases} \delta^r \text{ si } \forall i \in [n],\ x_i \geq \alpha_i^r, \\ 0 \text{ sinon.} \end{cases}$$

Pour tout select-set R on définit la fonction $f_R : \mathbf{X} \to L$ par

$$f_R(\mathbf{x}) = \bigvee_{r \in R} f_r(\mathbf{x}).$$

Pour toute règle de rejet r, on définit la fonction $f_r : \mathbf{X} \to L$ par

$$f_r(x_1, \ldots, x_n) = \begin{cases} \delta^r \text{ si } \forall i \in [n],\ x_i \leq \alpha_i^r, \\ 1 \text{ sinon.} \end{cases}$$

Pour tout reject-set R on définit la fonction $f_R : \mathbf{X} \to L$ par

$$f_R(\mathbf{x}) = \bigwedge_{r \in R} f_r(\mathbf{x}).$$

Toute fonction non-décroissante de $\mathbf{X}$ vers L peut être exprimée par un select-set et par un reject-set. On dit qu'un ensemble de règles R est équivalent à une fonction f si $f_R = f$.

Soient deux select-sets (resp. reject sets) R et R'. On dit que R et R' sont *équivalents* si $f_R = f_{R'}$, et on appelle $\mathrm{Eq}^{\vee}(R)$ (resp. $\mathrm{Eq}^{\wedge}(R)$) l'ensemble des select-sets (resp. reject-sets) équivalents à R. L'existence d'ensembles de règles équivalents est due au fait que certaines règles apportent plus d'informations que d'autres.

Pour deux règles de sélection r et s telles que

$$\delta^s \leq \delta^r \text{ et } \forall i \in [n],\ \alpha_i^r \leq \alpha_i^s$$

on a $r \Rightarrow s$. Notez que $r \Rightarrow s$ est équivalent à $f_s \leq f_r$ et à $f_{\{r,s\}} = f_{\{r\}}$.

Pour deux règles de rejet r et s telles que

$$\delta^r \leq \delta^s \text{ et } \forall i \in [n],\ \alpha_i^s \leq \alpha_i^r),$$

on a $r \Rightarrow s$. Notez que $r \Rightarrow s$ est équivalent à $f_r \leq f_s$ et à $f_{\{r,s\}} = f_{\{r\}}$.

Pour deux règles r et s telles que $r \Rightarrow s$, on dit que s est *redondante* par rapport à r. Soit un ensemble de règles R. On dit que R est *non redondant* si $r \Rightarrow s$ n'est vérifiée pour aucune paire de règles $r, s \in R$. On définit l'ensemble R^* par

$$R^* = \left\{ s \in R \;\middle|\; \forall r \in R,\ \left[r \not\Rightarrow s \text{ ou } s = r \right] \right\}.$$

La proposition suivante est prouvée dans l'Annexe B.1.

Proposition 63. Pour tout select-set (resp. reject-set) R, R^* est le plus petit élément de $\mathrm{Eq}^{\vee}(R)$ (resp. $\mathrm{Eq}^{\wedge}(R)$).

Apprentissage de règles de décision

Le plus ancien algorithme d'apprentissage de règles de décision qui garantit la monotonie du modèle est OLM [9]. Depuis, de nombreux algorithmes d'apprentissage d'ensembles de règles de décision ont été proposés, pour la plupart basés sur la théorie des rough sets et la DRSA. Parmi ces méthodes d'apprentissage de règles, nous retenons VC-DomLEM, car cet algorithme a été testé sur un ensemble de 12 jeux de données monotones et a montré commettre peu d'erreurs de prédiction par rapport aux modèles auxquels il a été comparé (voir [11, 19]). Dans le chapitre suivant, nous utiliserons cette méthode comme point de comparaison, en testant nos méthodes sur les mêmes jeux de données.

Une approche basée sur les rough sets : VCDomLEM

VC-DomLEM [19] est un algorithme d'apprentissage de règles de décision, qui peut être appliqué à la tache de classification classique et à la classification monotone. Dans sa version monotone, VC-DomLEM est basé sur une version de la DRSA probabiliste présentée dans [18]. Il dépend d'une mesure de consistance et de seuils $\theta_1, \ldots, \theta_{|L|}$ (qui sont des hyperparamètres de l'algorithme) par rapport auxquels il calcule des estimations basses des ensembles $C_1^{\geq}, \ldots, C_{|L|}^{\geq}$ (voir equation (5.2)). Ces estimations basses contiennent les observations servant de base à l'apprentissage de règles de décision. Pour chaque classe $k \in L$, l'algorithme construit un ensemble de règles R_k tel que $f_{R_k}(\mathbf{x}) = k$ pour tout $(\mathbf{x}, k) \in \underline{C_k^{\geq}}$. Cet ensemble de règles est obtenu par un algorithme basé sur le principe de *sequential covering* (aussi appelé *divide and conquer*) [47]. Les règles qui sont ajoutées itérativement au modèle en construction sont créées avec un nombre d'attributs actifs aussi faible que possible, afin de limiter le sur-apprentissage et de favoriser l'interprétabilité du modèle.

5.3.5 Règles de décision et boosting

Certaines méthodes comme MORE [37] ou LPRules [70] consistent en l'apprentissage d'une combinaison linéaire de règles à travers la méthode de boosting. Nous ne détaillons pas ici les méthodes de boosting utilisées, et nous contentons d'illustrer ce type de modèle en décrivant la forme des classifieurs appris par LPRules.

L'algorithme LPRules décompose le problème de classification monotone en $k-1$ problèmes de classification binaire dont on dénote les classifieurs par $f_1, \ldots, f_{k-1}$. Pour chaque problème de classification binaire, les classes considérées sont 0 et 1. Chaque règle Φ est une fonction définie par une condition de la forme

$$x_1 \geq \alpha_1, \ldots, x_n \geq \alpha_n \qquad \Big(\text{resp.} \quad x_1 \leq \alpha_1, \ldots, x_n \leq \alpha_n\Big)$$

telle que $\Phi(\mathbf{x})$ vaut 1 (resp. -1) si la conditions de Φ est vérifiée par $\mathbf{x}$, et 0 sinon. Le classifieur binaire correspondant à un ensemble de règles $\{\Phi_1, \ldots, \Phi_t\}$ est obtenu par une combinaison linéaire de ces règles :

$$f_i(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{t} a_j \Phi_j(\mathbf{x})\right),$$

où sign est la fonction qui vaut 1 si son argument et positif, 0 sinon, et $a_1, \ldots a_t$ sont des poids associés à chaque règle. Enfin, le classifieur final est défini comme la somme des $k-1$ classifieurs binaires

$$f(\mathbf{x}) = 1 + \sum_{i=1}^{k-1} f_i(\mathbf{x}).$$

La précision de ces méthodes tend à surpasser celle des ensembles de règles « classiques » que nous avons présentés dans la sous-section précédente. Cependant, l'interprétabilité associée à ces ensembles de règles (en particulier, l'interprétabilité ponctuelle) de décision est perdue, puisque les prédictions du classifieur sont le résultat d'une combinaison linéaire des différentes règles qui constituent le modèle, et que, par conséquent, une prédiction du modèle ne peut jamais être expliquée par une règle unique. Dans la suite, nous ne traitons pas de ce type de modèle.

5.4 Les intégrales floues pour la classification monotone

Les intégrales de Sugeno et de Choquet appartiennent à la classe des intégrales dites *floues*, et sont des fonctions d'agrégation reposant sur la notion de capacité. Ces fonctions ont, en particulier, été étudiées en tant que fonctions d'utilité dans le domaine de la MCDA.

Nous commençons par rappeler les notions de capacité et d'anti-capacité.

5.4.1 Capacité et anti-capacité

Soit L un ensemble ordonné borné dont le plus petit et le plus grand élément sont dénotés par 0 et 1, respectivement. Une fonction $\mu : 2^{[n]} \to L$ est une *capacité* si

$$\forall I \subseteq J \subseteq [n], \quad \mu(I) \leq \mu(J)$$

et si $\mu(\emptyset) = 0$ et $\mu([n]) = 1$. Pour tout $I \subseteq [n]$ on peut interpréter $\mu(I)$ comme étant la valeur d'importance du sous-ensemble d'attributs I.

Une fonction $\nu : 2^{[n]} \to L$ est une *anti-capacité* si

$$\forall I \subseteq J \subseteq [n], \quad \nu(I) \geq \nu(J)$$

et si $\mu(\emptyset) = 1$ et $\mu([n]) = 0$.

Pour toute capacité μ, il existe une anti-capacité duale ν telle que $\nu(I) = \mu([n] \backslash I)$ pour tout $I \subseteq [n]$.

5.4.2 Les intégrales de Choquet

Les intégrales de Choquet sont des fonctions d'agrégation dans $[0, 1]$ qui généralisent la notion de moyenne arithmétique pondérée, en accordant un poids à chaque sous-ensemble d'arguments. Le poids de chaque sous-ensemble est déterminé par la transformée de Möbius d'une capacité.

Soit $\mu : 2^{[n]} \to L$ une capacité. La *transformée de Möbius* de μ est la fonction $m_\mu : 2^{[n]} \to L$ définie par

$$m_\mu(I) = \sum_{J \subseteq I} (-1)^{|I|-|J|} \mu(J).$$

L'intégrale de Choquet définie par rapport à μ est la fonction $C_\mu : L^n \to L$ définie par

$$C_\mu(\mathbf{x}) = \sum_{I \subseteq [n]} m_\mu(I) \bigwedge_{i \in I} x_i. \tag{5.5}$$

Tandis que $\mu(I)$ peut être interprétée comme l'importance totale de l'ensemble I, $m_\mu(I)$ peut être vue comme le poids accordé spécifiquement à I. Notez que ce poids peut être positif (il modélise alors un phénomène de synergie des attributs de I), ou négatif (il modélise alors un phénomène de redondance).

Les intégrales de Choquet ont pour avantage d'être plus expressives que les moyennes arithmétiques pondérées, tout en étant non-décroissantes et globalement interprétables. Des outils mathématiques comme la valeur de Shapley [88], qui résume l'importance de chaque attribut par une valeur unique, et l'indice d'interaction, qui exprime un degré d'interaction de chaque paire d'attributs, permettent d'obtenir une compréhension globale d'une intégrale de Choquet au travers de quelques valeurs quantitatives.

Des modèles de classification monotone peuvent être définis sur la base de l'intégrale de Choquet [93], par exemple via une généralisation du principe de régression logistique [96] où le modèle linéaire est remplacé par une intégrale de Choquet. La *régression Choquistique* ainsi définie obtient des résultats de prédiction compétitifs avec les modèles auxquels elle est comparée (voir [93]).

5.4.3 Les intégrales de Sugeno

Les intégrales de Sugeno sont parfois vues comme les analogues ordinales des intégrales de Choquet. L'intégrale de Sugeno définie par rapport à μ est la fonction $S_\mu^\vee : L^n \to L$ définie par

$$S_\mu^\vee(\mathbf{x}) = \bigvee_{I \subseteq [n]} \mu(I) \wedge \bigwedge_{i \in I} x_i.$$

Toute intégrale de Sugeno peut également être exprimée en fonction d'une *anti-capacité*. Soit une anti-capacité $\nu : 2^{[n]} \to L$. L'intégrale de Sugeno définie par rapport à ν est la fonction $S_\nu^\wedge : L^n \to L$ définie par

$$S_\nu^\wedge(\mathbf{x}) = \bigwedge_{I \subseteq [n]} \nu(I) \vee \bigvee_{i \in I} x_i.$$

Notez que deux intégrales de Sugeno $S^{\vee}_{\mu}$ et $S^{\wedge}_{\nu}$ sont égales si et seulement si $\nu(I) = \mu([n]\backslash I)$ pour tout $I \subseteq [n]$.

Les *ensembles focaux* de μ (resp. de ν) sont les ensembles $F \subseteq [n]$ tels que

$$\mu(F) > \bigvee_{I \subset F} \mu(I) \qquad \left(\text{resp.} \quad \nu(F) < \bigwedge_{I \subset F} \nu(I)\right).$$

On dénote par $\mathcal{F}(\mu)$ (resp. par $\mathcal{F}(\nu)$) l'ensemble des ensembles focaux de μ (resp. de ν). Les valeurs de μ et ν sur tous les sous-ensembles de $[n]$ sont déterminées par leurs valeurs sur leurs ensembles focaux :

$$S^{\vee}_{\mu}(\mathbf{x}) = \bigvee_{I \in \mathcal{F}(\mu)} \mu(I) \wedge \bigwedge_{i \in I} x_i \quad \text{et} \quad S^{\wedge}_{\nu}(\mathbf{x}) = \bigwedge_{I \in \mathcal{F}(\nu)} \nu(I) \vee \bigvee_{i \in I} x_i.$$

5.4.4 Fonctions d'Utilité de Sugeno

Les intégrales de Sugeno sont des fonctions d'agrégation permettant la fusion de valeurs sur une échelle unique. Les Fonctions d'Utilité de Sugeno (FUS) généralisent la notion d'intégrale de Sugeno de manière à permettre la fusion de valeurs venant d'échelles différentes. Elles semblent avoir été considérées pour la première fois dans [58]. Les auteurs de [28, 29] définissent les FUS comme étant les fonctions composées d'une intégrale de Sugeno et d'une fonction $\varphi_i : X_i \to L$ sur chaque paramètre, qui vérifie les conditions

$$\varphi_i(0) = 0 \quad \text{et} \quad \varphi_i(1) = 1.$$

Ici, nous considérons des Fonctions d'Utilité de Sugeno pour lesquelles $\varphi_1, \ldots, \varphi_n$ sont des fonctions non décroissantes (voir [25]).

Une fonction $\varphi_i : X_i \to L$ est appelée *normalisation qualitative locale* (NQ locale) si elle est non-décroissante et si elle vérifie $\varphi_i(0) = 0$ et $\varphi_i(1) = 1$. Une fonction $\varphi : \mathbf{X} \to L^n$ est appelée *normalisation qualitative globale* (NQ globale) si elle peut être définie par le produit de n NQ locales, c'est à dire par

$$\varphi(\mathbf{x}) = (\varphi_1(x_1), \ldots, \varphi_n(x_n))$$

où, pour chaque $i \in [n]$, la fonction $\varphi_i : X_i \to L$ est une NQ locale. Remarquez que les NQ locales qui définissent une NQ globale sont uniques. Par conséquent, pour toute NQ globale $\varphi : \mathbf{X} \to L$, on dénotera par $\varphi_1, \ldots, \varphi_n$ les NQ locales dont le produit est φ.

Nous appelons *Fonction d'Utilité de Sugeno (FUS)* de $\mathbf{X}$ à L la composition d'une intégrale de Sugeno de L^n vers L avec une NQ globale de $\mathbf{X}$ vers L^n. Soit une capacité $\mu : 2^{[n]} \to L$ et une NQ globale $\varphi : \mathbf{X} \to L$. La FUS définie par rapport à μ et à φ est la fonction $S^{\vee}_{\mu,\varphi} : \mathbf{X} \to L$ définie par

$$\begin{aligned} S^{\vee}_{\mu,\varphi}(x_1, \ldots, x_n) &= S^{\vee}_{\mu}\big(\varphi_1(x_1), \ldots, \varphi_n(x_n)\big) \\ &= \bigvee_{I \subseteq [n]} \mu(I) \wedge \bigwedge_{i \in I} \varphi_i(x_i). \end{aligned}$$

Soit une anti-capacité $\nu : 2^{[n]} \to L$. La FUS définie par rapport à ν et à φ est la fonction $S^{\wedge}_{\nu,\varphi} : \mathbf{X} \to L$ définie par

$$\begin{aligned} S^{\wedge}_{\nu,\varphi}(x_1, \ldots, x_n) &= S^{\wedge}_{\nu}\big(\varphi_1(x_1), \ldots, \varphi_n(x_n)\big) \\ &= \bigwedge_{I \subseteq [n]} \nu(I) \vee \bigvee_{i \in I} \varphi_i(x_i). \end{aligned}$$

Remarque 64. Toutes les intégrales de Sugeno sont des FUS (où $X_1 = \cdots = X_n = L$ et où $\varphi_1, \ldots, \varphi_n$ sont des fonctions identités).

5.4.5 Interprétabilité des FUS

Les intégrales de Sugeno étant exprimées à l'aide des opérations $\wedge$ et $\vee$, elles peuvent aisément être décomposées en un ensemble de règles de décision [40] ; cette possibilité est conservée dans le cas des FUS. Nous verrons dans le chapitre suivant comment toute FUS peut être exprimée par un ensemble de règles de décision.

Chaque prédiction réalisée par une FUS peut donc être justifiée par une règle de décision. De ce fait, l'interprétabilité ponctuelle d'une FUS est au moins aussi élevée que celle de l'ensemble de règles de décision équivalent.

On peut en revanche avancer qu'une FUS, présentée comme la composition de NQ locales et d'une capacité, possède une interprétabilité globale plus importante que l'ensemble de règles équivalent, en particulier, si cet ensemble contient beaucoup de règles. Notez que cette hypothèse n'a, à notre connaissance, pas été testée empiriquement et est certainement modulée par plusieurs paramètres : la taille de l'ensemble de règles, mais aussi le nombre d'éléments focaux, les préférences personnelles de l'utilisateur, etc.

5.4.6 Expressivité des FUS

Illustrons le fonctionnement des FUS par un exemple simple.

Exemple 65. On considère $L = \{0, 1\}$, $\mathbf{X} = X_1 \times X_2$, avec $X_1 = X_2 = \{0, a, b, c, 1\}$ et $0 < a < b < c < 1$. On définit les NQ locales $\varphi_1 : X_1 \to L$ et de $\varphi_2 : X_2 \to L$ par

$$\varphi_1(x_1) = \begin{cases} 0 \text{ si } x_1 \leq a, \\ 1 \text{ sinon} \end{cases} \qquad \text{et} \quad \varphi_2(x_2) = \begin{cases} 0 \text{ si } x_2 \leq a, \\ 1 \text{ sinon.} \end{cases}$$

Une fois les fonctions φ_1 et φ_2 spécifiées, la définition de $S^{\vee}_{\mu,\varphi}$ dépend du choix des valeurs de μ, comme illustré par la Figure 5.3.

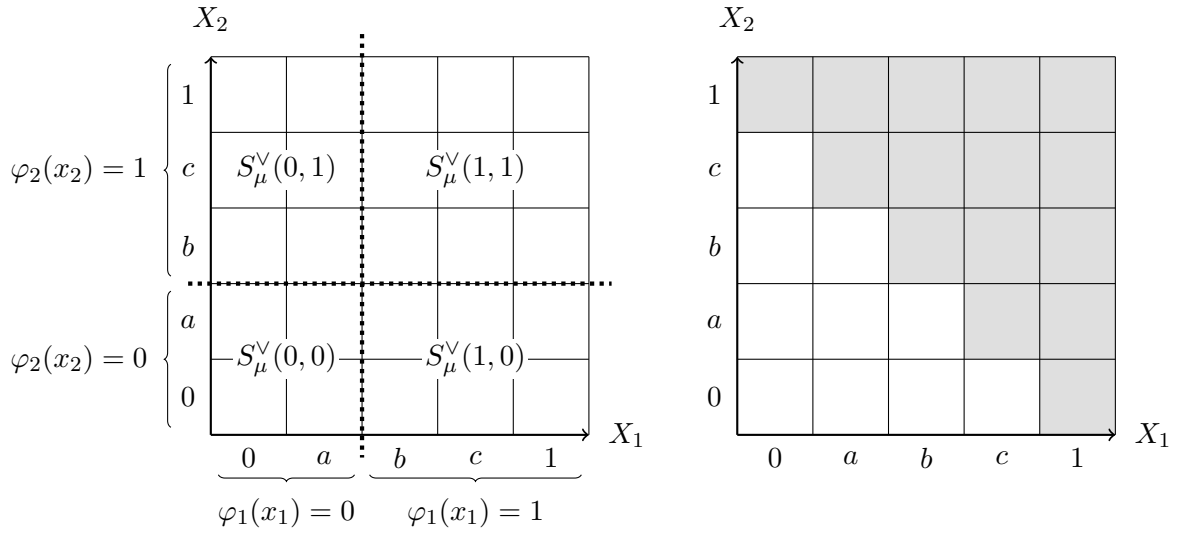


FIGURE 5.3 – Le schéma à gauche représente les NQ locales φ_1 et φ_2. Celles-ci découpent l'espace $\mathbf{X}$ en 4 parties, qui correspondent chacune à un point de L^2, sur lequel s'applique l'intégrale de Sugeno $S^\vee_\mu$. À droite, une fonction non décroissante « en escalier », qui est un exemple typique de fonction ne correspondant à aucune FUS.

Cette présentation visuelle montre notamment que certaines fonctions non-décroissantes ne sont pas représentables par une FUS.

L'exemple précédent met en évidence le fait que les FUS constituent une classe de fonctions plus restreinte que les fonctions non-décroissantes de $\mathbf{X}$ à L. En fait, les FUS souffrent d'un certain nombre de limitations. Remarquez d'abord que, pour une FUS $S^\vee_{\mu,\varphi}$, la NQ globale φ associe à chaque description $\mathbf{x} \in \mathbf{X}$ une valeur $\varphi(\mathbf{x}) \in L^n$, et on a

$$S^\vee_{\mu,\varphi}(\mathbf{x}) = S^\vee_\mu(\varphi(\mathbf{x})).$$

D'une certaine manière, L^n joue le rôle d'espace de description intermédiaire. Or la taille de L^n est imposée par le nombre de classes, ce qui est particulièrement restrictif lorsque L est de petite taille ; en particulier dans le cas de la classification binaire, comme illustré par l'exemple précédent.

Notez que même dans les cas où L est arbitrairement grand, les fonctions non décroissantes « en escalier » ne correspondent à aucune FUS. Cela est mis en évidence par la caractérisation des FUS suivante (voir [25, 31]) :

Proposition 66. Une fonction non décroissante $f : \mathbf{X} \to L$ est une FUS si et seulement si elle vérifie, pour tout $\mathbf{x}, \mathbf{x'} \in \mathbf{X}$, $i \in [n]$ et $a \in X_i$:

$$f(\mathbf{x}|_i^0) < f(\mathbf{x}|_i^a) \text{ et } f(\mathbf{x'}|_i^a) < f(\mathbf{x'}|_i^1) \quad \Longrightarrow \quad f(\mathbf{x}|_i^a) < f(\mathbf{x'}|_i^a)$$

où $\mathbf{x}|_i^a$ désigne le tuple dont toutes les composantes sont identiques à celles de $\mathbf{x}$, excepté la i-ème composante qui vaut a.

Afin de surmonter le manque d'expressivité des FUS, les auteurs de [23] introduisent un modèle qui consiste en un maximum (ou un minimum) de plusieurs FUS, dont l'intérêt est de permettre la modélisation de n'importe quel ensemble de données monotones. Dans le chapitre

suivant, nous étudions les FUS et les ensembles de FUS en nous appuyant sur le paradigme des règles de décision. Nous proposons notamment une méthode pour apprendre un ensemble de FUS à partir de données empiriques.

Chapitre 6

FUS et règles de décision pour la classification monotone

6.1 Préliminaires

Toute FUS peut être exprimée par un ensemble de règles de décision. À l'inverse, certains ensembles de règles ne peuvent pas être exprimés par une FUS. Dans le but de palier au manque d'expressivité des FUS, les auteurs de [23] introduisent une manière d'exprimer tout classifieur monotone respectant

$$f(0,\ldots,0)=0 \quad \text{et} \quad f(1,\ldots,1)=1$$

par le maximum ou par le minimum de plusieurs FUS. Cette formulation permet de considérer les ensembles de FUS comme des alternatives aux ensembles de règles de décision. La motivation principale des études de ce chapitre est de déterminer si ces alternatives peuvent être avantageuses en terme d'interprétabilité globale, tout en ayant une précision compétitive. En effet, on peut considérer qu'un classifieur basé sur un petit ensemble de FUS possède une meilleure interprétabilité globale qu'un classifieur basé sur un ensemble de règles de grande cardinalité. Nous étudions donc le nombre de FUS nécessaires afin de modéliser des données empiriques avec un précision acceptable. Nos études débouchent sur un algorithme d'apprentissage de classifieurs basés sur des ensembles de FUS, que nous appelons RL-SUF. Le chapitre est organisé de la manière suivante.

Nous clarifions d'abord, d'un point de vue formel, le lien entre les règles de décision et les FUS, en définissant des procédures permettant de « traduire » toute FUS en un ensemble de règles (Section 6.2), ainsi que des procédures permettant de traduire tout ensemble de règles pour lequel cela est possible en une unique FUS (Section 6.3). Nous présentons ensuite les modèles basés sur des ensembles de FUS, en relation avec les ensemble de règles de décision (Section 6.4). Dans la Section 6.5 nous définissons une méthode non-paramétrique d'apprentissage de règles qui sera utilisée dans les études empiriques des trois section suivantes. Comme résultat subsidiaire, nous montrons que cet algorithme est compétitif avec VC-DomLEM.

En nous appuyant sur les éléments introduits dans les sections précédentes, nous tentons de déterminer le nombre de FUS nécessaires à la réalisation de plusieurs tâches.

- représenter exactement un ensemble de règles de décision appris à partir de données empiriques (Section 6.6),
- interpoler un ensemble de données monotones (Section 6.7),

- apprendre un classifieur dont les prédictions sur les données de test sont compétitives avec VC-DomLEM (Section 6.8).

Les implémentations des algorithmes présentés dans ce chapitre, ainsi que les jeux de données utilisés durant nos tests, sont disponibles à l'adresse `https://github.com/QGBrabant/SUF4OC`.

Dans ce chapitre, nous utilisons les conventions et notations suivantes. L'ensemble $\mathbf{X}$ est le produit cartésien $X_1 \times \cdots \times X_n$, où chaque ensemble X_i est borné et totalement ordonné. L'ensemble L est un ensemble fini et totalement ordonné. Les capacités sont dénotées par μ et les anti-capacités sont dénotées par ν. L'intégrale de Sugeno définie par rapport à une capacité μ (resp. une anti-capacité ν) est dénotée par $S^{\vee}_{\mu}$ (resp. par $S^{\wedge}_{\nu}$). Toute NQ globale est dénotée par φ et est vue comme le produit de NQ locales dénotées par $\varphi_1, \ldots, \varphi_n$. La FUS définie par rapport à une capacité μ (resp. une anti-capacité ν) et une NQ globale φ est dénotée par $S^{\vee}_{\mu,\varphi}$ (resp. $S^{\wedge}_{\nu,\varphi}$). Les FUS dont les capacités/anti-capacités et NQ globales correspondantes ne sont pas spécifiées sont dénotées par S. L'ensemble des FUS de $\mathbf{X}$ à L est dénoté par $\mathcal{S}_{\mathbf{X},L}$, et l'ensemble des select-sets (resp. reject-sets) non-redondants décrivant des fonctions de $\mathbf{X}$ à L est dénoté par $\mathcal{R}^{\vee}_{\mathbf{X},L}$ (resp. $\mathcal{R}^{\wedge}_{\mathbf{X},L}$). Enfin, pour tout ensemble de règles R, l'ensemble non redondant équivalent à R sera dénoté par R^*.

6.2 Traduction des FUS en règles

Puisque les FUS sont des fonctions monotones, toute FUS peut être traduite par un select-set et par un reject-set. Plus précisément : pour toute FUS S, il existe exactement un select-set non redondant R tel que $S = f_R$ et un reject-set non redondant R' tel que $S = f_{R'}$.

Dans cette section on définit la fonction $\mathbb{S}$-$\mathbb{SET} : \mathcal{S}_{\mathbf{X},L} \to \mathcal{R}^{\vee}_{\mathbf{X},L}$, qui associe à chaque FUS un select-set non redondant équivalent, et la fonction $\mathbb{R}$-$\mathbb{SET} : \mathcal{S}_{\mathbf{X},L} \to \mathcal{R}^{\wedge}_{\mathbf{X},L}$, qui associe à chaque FUS un reject-set non redondant équivalent. Comme cela devrait apparaitre clairement dans les sous-sections qui suivent, une FUS définie par rapport à une capacité se traduit plus naturellement par un select-set tandis qu'une FUS définie par rapport à une anti-capacité se traduit plus naturellement par un reject-set. Nous donnons donc les définitions de $\mathbb{S}$-$\mathbb{SET}$ et $\mathbb{R}$-$\mathbb{SET}$ en terme de capacité et anti-capacité, respectivement.

6.2.1 Intégrale de Sugeno en règles de sélection

Soit $S^{\vee}_{\mu} : L^n \to L$ l'intégrale de Sugeno définie par rapport à la capacité μ. Cette intégrale de Sugeno vérifie

$$S^{\vee}_{\mu}(\mathbf{x}) = \bigvee_{F \in \mathcal{F}(\mu)} g^{\vee}_F(\mathbf{x}), \quad \text{où} \quad g^{\vee}_F(\mathbf{x}) = \mu(F) \wedge \bigwedge_{i \in F} x_i.$$

Voyons d'abord comment représenter chacune de ces fonctions $g^{\vee}_F$ par un ensemble de règles de sélection. Pour tout $\delta \in L$, l'inégalité

$$\mu(F) \wedge \bigwedge_{i \in F} x_i \geq \delta$$

est vérifiée si et seulement si

$$\mu(F) \geq \delta \quad \text{et} \quad \forall i \in F,\ x_i \geq \delta.$$

Par conséquent, l'ensemble minimal de règles de sélection qui exprime $g^{\vee}_F$ est

$$\mathbb{S}\text{-}\mathbb{SET}(g^{\vee}_F) = \Big\{ \big[\forall i \in F,\ x_i \geq \delta\big] \Rightarrow y \geq \delta \;\Big|\; \delta \in L,\ \delta \leq \mu(F) \Big\}.$$

et le select-set non redondant qui traduit $S_\mu^\vee$ est

$$\$\text{-}\mathbb{SET}(S_\mu^\vee) = \left(\bigcup_{F \in \mathcal{F}(\mu)} \$\text{-}\mathbb{SET}(g_F^\vee) \right)^*.$$

Rappel : pour tout ensemble de règles R, la notation R^* désigne le plus petit ensemble de règles équivalent.

6.2.2 Intégrale de Sugeno en règles de rejet

De manière duale, l'intégrale de Sugeno peut également être exprimée par des règles de rejet. Soit $S_\nu^\wedge : L^n \to L$ l'intégrale de Sugeno définie par rapport à l'anti-capacité ν. Cette intégrale de Sugeno vérifie

$$S_\nu^\wedge(\mathbf{x}) = \bigwedge_{F \in \mathcal{F}(\nu)} g_F^\wedge(\mathbf{x}), \quad \text{où} \quad g_F^\wedge(\mathbf{x}) = \nu(F) \vee \bigvee_{i \in F} x_i.$$

Pour tout $\delta \in L$, l'inégalité

$$\nu(F) \vee \bigvee_{i \in F} x_i \leq \delta$$

est vérifiée si et seulement si

$$\nu(F) \leq \delta \quad \text{et} \quad \forall i \in F,\ x_i \leq \delta.$$

Par conséquent, l'ensemble minimal de règles de rejet qui exprime $g_F^\wedge$ est

$$\mathbb{R}\text{-}\$\mathbb{ET}(g_F^\wedge) = \Big\{ \big[\forall i \in F,\ x_i \leq \delta\big] \Rightarrow y \leq \delta \;\Big|\; \delta \in L,\ \delta \geq \nu(F) \Big\}.$$

et le reject-set non redondant qui traduit $S_\nu^\wedge$ est

$$\mathbb{R}\text{-}\$\mathbb{ET}(S_\nu^\wedge) = \left(\bigcup_{F \in \mathcal{F}(\nu)} \mathbb{R}\text{-}\$\mathbb{ET}(g_F^\wedge) \right)^*.$$

6.2.3 FUS en règles

Pour toute capacité μ et NQ globale φ, la FUS $S_{\mu,\varphi}^\vee$ vérifie

$$S_{\mu,\varphi}^\vee(\mathbf{x}) = S_\mu^\vee(\varphi_1(x_1), \ldots, \varphi_n(x_n))$$

pour tout $\mathbf{x} \in L$. Par conséquent, $S_{\mu,\varphi}^\vee$ peut être exprimée par l'ensemble de règles

$$\bigcup_{F \in \mathcal{F}(\mu)} \Big\{ \big[\forall i \in F,\ \varphi_i(x_i) \geq \delta\big] \Rightarrow y \;\Big|\; \delta \in L,\ \delta \leq \mu(F) \Big\}.$$

Cet ensemble n'est pas à proprement parler un select-set, puisque ses règles sont exprimées en fonction des valeurs de $\varphi_1(x_1), \ldots, \varphi_n(x_n)$. Cependant, pour tout $i \in [n]$, l'inégalité

$$\varphi_i(x_i) \geq \delta$$

est équivalente à

$$x_i \geq \bigwedge \big\{ a_i \in X_i \;\big|\; \varphi_i(a_i) \geq \delta \big\}.$$

Le select-set non redondant qui exprime $S^{\vee}_{\mu,\varphi}$ est donc

$$\$\text{-}\mathbb{SET}(S^{\vee}_{\mu,\varphi}) = \left(\bigcup_{F \in \mathcal{F}(\mu)} \left\{ \left[\forall i \in F,\ x_i \geq \alpha^{\vee}(i,\delta) \right] \Rightarrow y \geq \delta \;\middle|\; \delta \in L,\ \delta \leq \mu(F) \right\} \right)^{*}.$$

où $\alpha^{\vee}(i,\delta)$ est défini par $\alpha^{\vee}(i,\delta) = \bigwedge \{ a_i \in X_i \mid \varphi_i(a_i) \geq \delta \}$. De manière duale, le reject-set non redondant traduisant une FUS $S^{\vee}_{\mu,\varphi}$ est donné par

$$\mathbb{R}\text{-}\$\mathbb{SET}(S^{\wedge}_{\nu,\varphi}) = \left(\bigcup_{F \in \mathcal{F}(\nu)} \left\{ \left[\forall i \in F,\ x_i \leq \alpha^{\wedge}(i,\delta) \right] \Rightarrow y \leq \delta \;\middle|\; \delta \in L,\ \delta \geq \nu(F) \right\} \right)^{*}.$$

où $\alpha^{\wedge}(i,\delta)$ est défini par $\alpha^{\wedge}(i,\delta) = \bigvee \{ a_i \in X_i \mid \varphi_i(a_i) \leq \delta \}$.

Remarque 67. La démonstration que $S^{\vee}_{\mu,\varphi} = f_{\$\text{-}\mathbb{SET}(S^{\vee}_{\mu,\varphi})}$ et que $S^{\wedge}_{\nu,\varphi} = f_{\mathbb{R}\text{-}\$\mathbb{SET}(S^{\wedge}_{\nu,\varphi})}$ sera incluse dans la démonstration de la Proposition 69 (voir Annexes B.2).

6.3 Traduction des règles en FUS

On dit qu'un ensemble de règles R est *FUS-représentable* s'il existe une FUS S équivalente à R (c'est à dire telle que $S = f_R$). Nous définissons les fonctions,

$$\mathbb{FUS}^{\vee} : \mathcal{R}^{\vee}_{\mathbf{X},L} \to \mathcal{S}_{\mathbf{X},L} \quad \text{et} \quad \mathbb{FUS}^{\wedge} : \mathcal{R}^{\wedge}_{\mathbf{X},L} \to \mathcal{S}_{\mathbf{X},L}$$

qui associent une FUS à chaque select-set et chaque reject-set, respectivement. Nous montrons ensuite que si R est un select-set (resp. reject-set) FUS-représentable alors $\mathbb{FUS}^{\vee}(R^*)$ (resp. $\mathbb{FUS}^{\wedge}(R^*)$) est équivalent à R. On définit les fonctions $\mathbb{FUS}^{\vee}$ et $\mathbb{FUS}^{\wedge}$ de la manière suivante.

Si R est un select-set non redondant,

$$\mathbb{FUS}^{\vee}(R) = S^{\vee}_{\mu,\varphi}$$

où μ est la capacité définie par

$$\mu(I) = \bigvee \{ \delta^r \mid r \in R,\ A^r \subseteq I \}, \tag{6.1}$$

pour tout $\emptyset \subset I \subset [n]$, et où φ est la NQ globale définie par

$$\varphi_i(x_i) = \bigvee \{ \delta^r \mid r \in R,\ 0 < \alpha^r_i \leq x_i \} \tag{6.2}$$

pour tout $i \in [n]$ et $x_i \in X_i \backslash \{0,1\}$. Notez qu'on a nécessairement $\mu(\emptyset) = 0$, $\mu([n]) = 1$, et $\varphi_i(0) = 0$ et $\varphi_i(1) = 1$ pour chaque $i \in [n]$.

Si R est un reject-set non redondant,

$$\mathbb{FUS}^{\wedge}(R) = S^{\wedge}_{\nu,\varphi}$$

où ν est l'anti-capacité définie par

$$\nu(I) = \bigwedge \{ \delta^r \mid r \in R,\ A^r \subseteq I \} \tag{6.3}$$

pour tout $\emptyset \subset I \subset [n]$, et où φ est la NQ globale définie par

$$\varphi_i(x_i) = \bigwedge \{ \delta^r \mid r \in R,\ x_i \leq \alpha^r_i < 1 \} \tag{6.4}$$

pour tout $i \in [n]$ et $x_i \in X_i \backslash \{0,1\}$. Notez qu'on a nécessairement $\nu(\emptyset) = 1$, $\nu([n]) = 0$, et $\varphi_i(0) = 0$ et $\varphi_i(1) = 1$ pour chaque $i \in [n]$.

De manière plus informelle, on peut également voir $\mathbb{FUS}^{\vee}(R)$ et $\mathbb{FUS}^{\wedge}(R)$ comme étant la FUS définie itérativement via les étapes ci-dessous.

Si R est un select-set.

1. On pose
$$\mu(I) = \begin{cases} 1 \text{ si } I = [n], \\ 0 \text{ sinon}, \end{cases}$$
$$\forall i \in [n],\ \varphi_i(x_i) = \begin{cases} 1 \text{ si } x_i = 1, \\ 0 \text{ sinon}. \end{cases}$$
La FUS $S^{\vee}_{\mu,\varphi}$ est la plus petite FUS de $\mathbf{X}$ vers L.
2. Pour chaque règle $r \in R$
   (a) on augmente la valeur de $\mu(A^r)$ jusqu'à δ^r,
   (b) pour chaque $i \in A^r$, on augmente $\varphi_i(\alpha_i)$ jusqu'à δ^r.

Si R est un reject-set.

1. On pose
$$\nu(I) = \begin{cases} 0 \text{ si } I = [n], \\ 1 \text{ sinon}, \end{cases}$$
$$\forall i \in [n],\ \varphi_i(x_i) = \begin{cases} 0 \text{ si } x_i = 0, \\ 1 \text{ sinon}. \end{cases}$$
La FUS $S^{\wedge}_{\nu,\varphi}$ est la plus grande FUS de $\mathbf{X}$ vers L.
2. Pour chaque règle $r \in R$
   (a) on diminue la valeur de $\nu(A^r)$ jusqu'à δ^r,
   (b) pour chaque $i \in A^r$, on diminue $\varphi_i(\alpha_i)$ jusqu'à δ^r.

Remarque 68. Dans la procédure ci-dessus, l'expression « augmenter $f(x)$ jusqu'à y » signifie que, pour tout x' appartenant au domaine de f tel que $x' \geq x$, si $f(x') < y$ alors $f(x')$ prend pour nouvelle valeur y. L'expression « diminuer $f(x)$ jusqu'à y » est à comprendre de manière duale.

Notez que pour tout select-set R on a $\mathbb{FUS}^{\vee}(R^*) \geq R$, tandis que pour tout reject-set R on a $\mathbb{FUS}^{\wedge}(R^*) \leq R$. La démonstration de la proposition suivante ainsi que de son Corollaire sont donnés dans l'Annexe B.2.

Proposition 69. Pour toute FUS S on a
$$\begin{aligned} S &= f_{\mathbb{S}\text{-}\mathbb{SET}(S)} = \mathbb{FUS}^{\vee}(\mathbb{S}\text{-}\mathbb{SET}(S)) \\ &= f_{\mathbb{R}\text{-}\mathbb{SET}(S)} = \mathbb{FUS}^{\wedge}(\mathbb{R}\text{-}\mathbb{SET}(S)). \end{aligned}$$

Corollaire 70. Un select-set (resp. reject set) R est FUS-représentable si et seulement si $\mathbb{FUS}^{\vee}(R^*) = f_R$ (resp. $\mathbb{FUS}^{\wedge}(R^*) = f_R$).

Ce corollaire permet de vérifier si un select-set R (resp. reject-set) est FUS-représentable en testant si $R^* = \mathbb{S}\text{-}\mathbb{SET}(\mathbb{FUS}^{\vee}(R^*))$ (resp. $R^* = \mathbb{R}\text{-}\mathbb{SET}(\mathbb{FUS}^{\wedge}(R^*))$).

Exemple 71. Considérons $L = \{0, 1\}$, $X_1 = X_2 = \{0, a, b, 1\}$ avec $0 < a < b < 1$, ainsi que les règles $r_1 = \big[(a, b) \nearrow 1\big]$ et $r_2 = \big[(b, a) \nearrow 1\big]$ et calculons φ_1, φ_2 et μ tels que $\mathbb{FUS}^{\vee}(\{r_1, r_2\}) = S^{\vee}_{\mu,\varphi}$. La règle r_1 implique
$$\varphi_1(a) \geq 1 \quad \text{et} \quad \varphi_2(b) \geq 1.$$
La règle r_2 implique
$$\varphi_1(b) \geq 1 \quad \text{et} \quad \varphi_2(a) \geq 1.$$
Les deux règles impliquent $\mu(\{1, 2\}) \geq 1$. On a donc
$$\varphi_1(a) = 1, \quad \varphi_2(a) = 1 \quad \text{et} \quad \mu(\{1, 2\}) = 1.$$
Les fonctions $f_{\{r_1,r_2\}}$ et $\mathbb{FUS}^{\vee}(\{r_1, r_2\})$ sont représentées dans la Figure 6.1. La traduction de $\mathbb{FUS}^{\vee}(\{r_1, r_2\})$ en règles est l'ensemble $\{(a, a) \nearrow 1\}$. On a $\mathbb{FUS}^{\vee}(\{r_1, r_2\}) > f_{\{r_1,r_2\}}$, donc $\{r_1, r_2\}$ n'est pas FUS-représentable.

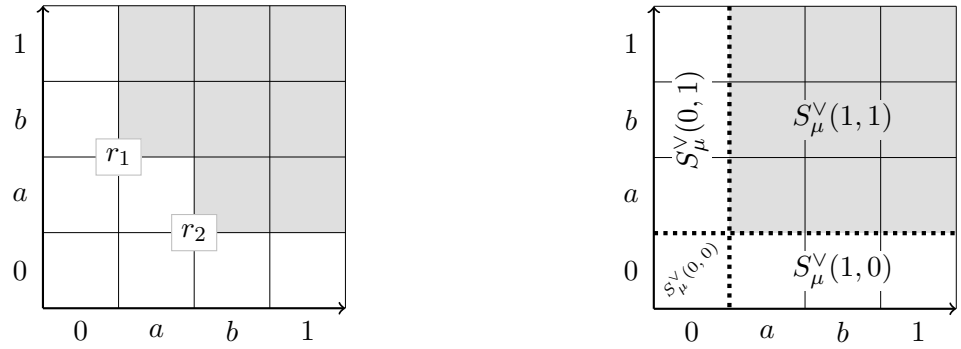


FIGURE 6.1 – Représentations de la fonction définie par les règles $r_1 = \big[(a,b) \nearrow 1\big]$ et $r_2 = \big[(b,a) \nearrow 1\big]$ (à gauche) et de la fonction $\mathbb{FUS}^{\vee}(\{r_1, r_2\})$ (à droite). Les lignes pointillées représentent le découpage de $\mathbf{X}$ induit par les fonctions φ_1 et φ_2, d'une manière similaire à celle de la Figure 5.3.

Pour tout select-sets non redondants R et R' tels que $R \subseteq R'$, on a $\mathbb{FUS}^{\vee}(R) \leq \mathbb{FUS}^{\vee}(R')$. Pour tout reject-sets R et R' tels que $R \subseteq R'$, on a $\mathbb{FUS}^{\wedge}(R') \leq \mathbb{FUS}^{\wedge}(R)$. En revanche, pour deux select-sets (resp. reject-sets) R et R', il est possible que $f_R < f_{R'}$ tandis que $\mathbb{FUS}^{\vee}(R') < \mathbb{FUS}^{\vee}(R)$ (resp. $\mathbb{FUS}^{\wedge}(R') < \mathbb{FUS}^{\wedge}(R)$), comme le montre l'exemple suivant.

Exemple 72. Considérons $X_1 = X_2 = L = \{0, a, b, c, 1\}$, ainsi que les select-sets non-redondants

$$R = \{(b,a) \nearrow 1,\ (0,b) \nearrow 1,\ (c,0) \nearrow 1\}$$

et

$$R' = \{(b,0) \nearrow 1,\ (0,b) \nearrow 1\}.$$

Remarquez que $f_R < f_{R'}$. On appelle $S^{\vee}_{\mu,\varphi}$ la FUS égale à $\mathbb{FUS}^{\vee}(R)$. On a $\mu(\{1\}) = \mu(\{2\}) = 1$ et

x	0	a	b	c	1
$\varphi_1(x)$	0	0	1	1	1
$\varphi_2(x)$	0	1	1	1	1

On appelle $S^{\vee}_{\mu',\varphi'}$ la FUS égale à $\mathbb{FUS}^{\vee}(R')$. On a $\mu'(\{1\}) = \mu'(\{2\}) = 1$ et

x	0	a	b	c	1
$\varphi'_1(x)$	0	0	1	1	1
$\varphi'_2(x)$	0	0	1	1	1

On a donc $f_R < f_{R'}$ et $\mathbb{FUS}^{\vee}(R') < \mathbb{FUS}^{\vee}(R)$. Ce phénomène est dû aux égalités (6.2) et (6.4), qui définissent les NQ locales de manière à ignorer les attributs non actifs des règles. Or, la règle $(b,0) \nearrow 1$ possède moins d'attributs actifs que $(b,a) \nearrow 1$, bien que $f_{(b,a)\nearrow 1} < f_{(b,0)\nearrow 1}$.

6.4 Ensembles de FUS

Afin d'exprimer n'importe quelle fonction $f : \mathbf{X} \to L$ vérifiant

$$f(0,\ldots,0) = 0 \quad \text{et} \quad f(1,\ldots,1) = 1,$$

on peut faire appel au maximum ou au minimum de plusieurs FUS.

6.4.1 Ensembles de FUS

Soit $\mathbf{S}$ un ensemble de FUS. On appelle $\bigvee \mathbf{S}$ et $\bigwedge \mathbf{S}$ les fonctions définies par

$$\bigvee \mathbf{S}(\mathbf{x}) = \bigvee_{S \in \mathbf{S}} S(\mathbf{x}) \quad \text{et} \quad \bigwedge \mathbf{S}(\mathbf{x}) = \bigwedge_{S \in \mathbf{S}} S(\mathbf{x}).$$

La proposition suivante est démontrée dans [23].

Proposition 73. Pour toute fonction $f : \mathbf{X} \to L$ non décroissante vérifiant $f(0, \ldots, 0) = 0$ et $f(1, \ldots, 1) = 1$, il existe un ensemble de FUS $\mathbf{S}$ tel que $f = \bigvee \mathbf{S}$ et un ensemble de FUS $\mathbf{S}'$ tel que $f = \bigwedge \mathbf{S}'$.

6.4.2 FUS-couverture d'un ensemble de règles

Afin de déterminer un ensemble de FUS équivalent à un ensemble de règles donné, on passe par les notions de $\vee$-*FUS-couverture* et $\wedge$-*FUS-couverture*. Une famille $\mathbf{P}$ d'ensembles de règles est une *couverture* d'un ensemble de règles R si

$$\bigcup_{P \in \mathbf{P}} P = R.$$

Soit un select-set R. Une couverture $\mathbf{P}$ de R est une $\vee$-*FUS-couverture* de R si

$$\forall P \in \mathbf{P}, \quad \mathbb{FUS}^{\vee}(P) \leq f_R. \tag{6.5}$$

Pour toute $\vee$-FUS-couverture $\mathbf{P}$ de R on a,

$$\begin{aligned} f_R \geq \bigvee_{P \in \mathbf{P}} \mathbb{FUS}^{\vee}(P) &\geq \bigvee_{P \in \mathbf{P}} f_P \\ &= \bigvee_{P \in \mathbf{P}} \bigvee_{r \in P} f_r \\ &= f_R, \end{aligned}$$

Par conséquent l'ensemble de FUS

$$\mathbf{S} = \{\mathbb{FUS}^{\vee}(P) \mid P \in \mathbf{P}\}$$

vérifie $\bigvee \mathbf{S} = f_R$.

Soit un reject-set R. Une couverture $\mathbf{P}$ de R est une $\wedge$-*FUS-couverture* de R si

$$\forall P \in \mathbf{P}, \quad f_R \leq \mathbb{FUS}^{\wedge}(P). \tag{6.6}$$

Pour toute $\wedge$-FUS-couverture $\mathbf{P}$ de R on a,

$$\begin{aligned} f_R \leq \bigwedge_{P \in \mathbf{P}} \mathbb{FUS}^{\wedge}(P) &\leq \bigwedge_{P \in \mathbf{P}} f_P \\ &= \bigwedge_{P \in \mathbf{P}} \bigwedge_{r \in P} f_r \\ &= f_R. \end{aligned}$$

Par conséquent l'ensemble de FUS

$$\mathbf{S} = \{\mathbb{FUS}^{\wedge}(P) \mid P \in \mathbf{P}\}$$

vérifie $\bigwedge \mathbf{S} = f_R$.

6.5 Un algorithme d'apprentissage de règles : SRL

Dans les sections qui suivent, nous étudions l'utilisation des ensembles de FUS pour différentes tâches, et notamment pour la modélisation de données empiriques. Dans cette étude, nous faisons appel à un algorithme d'apprentissage (non paramétrique) d'ensemble de règles de décision, que nous définissons dans cette section. Cet algorithme se décline en deux variantes : SRL$^{\vee}$ retournant un select-set et SRL$^{\wedge}$, duale de la première, retournant un reject-set.

L'apprentissage est effectué à partir d'un ensemble de données $\mathcal{D}$. La première étape de l'algorithme consiste à effectuer un relabeling de $\mathcal{D}$ qui minimise la MAE. On appelle $\mathcal{M}$ l'ensemble de données monotone résultant de ce relabeling.

On dénote par R^- le select-set non redondant qui représente la fonction $\lambda^-_{\mathcal{M}}$, et par R^+ le reject-set non redondant qui représente la fonction $\lambda^+_{\mathcal{M}}$.

Les fonctions f_{R^-} et f_{R^+} sont des interpolations de $\mathcal{M}$. Cependant les règles de R^- et R^+ possèdent pour la plupart un grand nombre d'attributs actifs. Dans le but de limiter un éventuel sur-apprentissage et de construire un modèle aussi interprétable que possible, notre algorithme effectue une simplification de ces ensembles de règles. Soit une règle de décision r. On appelle *simplification* de r toute règle s telle que $\delta^s = \delta^r$, $A^s \subseteq A^r$ et $\forall i \in A^s$, $\alpha^s_i = \alpha^r_i$. Pour un ensemble de règles R, on appelle simplification de R tout ensemble de règles composé de simplifications des règles de R.

L'ensemble retourné par SRL$^\vee$ est une simplification de R^-, tandis que l'ensemble retourné par SRL$^\wedge$ est une simplification de R^+.

6.5.1 Simplification de R^-, simplification de R^+

On dit qu'une règle de sélection $\boldsymbol{\alpha} \nearrow \delta$ est compatible avec $\mathcal{M}$ si pour tout $(\mathbf{x}, y) \in \mathcal{M}$

$$\mathbf{x} \geq \boldsymbol{\alpha} \quad \Longrightarrow \quad y \geq \delta.$$

Pour chaque règle $r \in R^-$, on cherche un ensemble $A \subseteq A^r$ de taille minimale tel que la règle

$$\forall i \in A,\ x_i \geq \alpha^r_i \quad \Longrightarrow \quad y \geq \delta^r$$

est compatible avec $\mathcal{M}$. Pour toute règle de selection r et tout $I \subseteq [n]$, on dénote par $r\backslash I$ la règle définie par

$$r\backslash I = (\alpha_1, \dots \alpha_n) \nearrow \delta^r,$$

où, pour chaque $i \in [n]$,

$$\alpha_i = \begin{cases} 0 \text{ si } i \in I, \\ \alpha^r_i \text{ sinon.} \end{cases}$$

On dit qu'une règle de rejet $\boldsymbol{\alpha} \searrow \delta$ est compatible avec $\mathcal{M}$ si pour tout $(\mathbf{x}, y) \in \mathcal{M}$

$$\mathbf{x} \leq \boldsymbol{\alpha} \quad \Longrightarrow \quad y \leq \delta.$$

Pour chaque règle $r \in R^+$, on cherche un ensemble $A \subseteq A^r$ de taille minimale tel que la règle

$$\forall i \in A,\ x_i \leq \alpha^r_i \quad \Longrightarrow \quad y \leq \delta^r$$

est compatible avec $\mathcal{M}$. Pour toute règle de rejet r et tout $I \subseteq [n]$, on dénote par $r\backslash I$ la règle définie par

$$r\backslash I = (\alpha_1, \dots \alpha_n) \searrow \delta^r,$$

où, pour chaque $i \in [n]$,

$$\alpha_i = \begin{cases} 1 \text{ si } i \in I, \\ \alpha^r_i \text{ sinon.} \end{cases}$$

Autrement dit, la règle $r\backslash I$ correspond à la règle r dans laquelle les attributs de I ne sont plus actifs.

Pour chaque règle r dans R^- (resp., dans R^+), on cherche le plus grand ensemble $I \subseteq [n]$ tel que $r\backslash I$ est compatible avec $\mathcal{M}$. On effectue cette recherche de manière gloutonne en ajoutant itérativement des éléments à I, et en testant à chaque ajout si la règle $r\backslash I$ est compatible avec $\mathcal{M}$. Puisqu'il est possible que $r\backslash\{i\}$ et $r\backslash\{j\}$ soient compatibles avec $\mathcal{M}$ tandis que $r\backslash\{i, j\}$ ne l'est pas, le résultat dépend de l'ordre dans lequel les attributs sont considérés. On détermine cet ordre d'après une estimation de l'importance de chacun des seuils $\alpha_1, \dots, \alpha_n$ de la règle considérée.

Pour les règles de sélection, cette estimation est donnée par la fonction $u_i^\vee : X_i \times L \to \mathbb{N}$ définie par

$$u_i^\vee(\alpha_i, \delta) = \left|\left\{(\mathbf{x}, y) \in \mathcal{M} \;\middle|\; \left[y \geq \delta \text{ et } x_i \geq \alpha_i\right] \text{ ou } \left[y < \delta \text{ et } x_i < \alpha_i\right]\right\}\right|.$$

Pour les règles de rejet, cette estimation est donnée par la fonction $u_i^\wedge : X_i \times L \to \mathbb{N}$ définie par

$$u_i^\wedge(\alpha_i, \delta) = \left|\left\{(\mathbf{x}, y) \in \mathcal{M} \;\middle|\; \left[y \leq \delta \text{ et } x_i \leq \alpha_i\right] \text{ ou } \left[y > \delta \text{ et } x_i > \alpha_i\right]\right\}\right|.$$

La procédure de simplification des règles est formalisée par les Algorithmes 13 et 14.

Algorithme 13: Simplification de R^-.

Entrées: Un ensemble de données monotones $\mathcal{M}$ et un select-set R^- représentant $\lambda^-_{\mathcal{M}}$
Sorties: Un ensemble de règles compatibles avec $\mathcal{M}$

fonction SIMPLIFICATION$^\vee(\mathcal{M}, R^-)$
  $R \leftarrow \{\}$
  pour chaque $r \in R^-$ **faire**
    $I \leftarrow \{\}$
    pour $i \in [n]$ par ordre croissant de $u^\vee(\alpha_i, i)$ **faire**
      si $r \backslash (I \cup \{i\})$ est compatible avec $\mathcal{M}$ **alors**
        $I \leftarrow I \cup \{i\}$
    $R \leftarrow R \cup \{r \backslash I\}$
  retourner R^*

Algorithme 14: Simplification de R^+.

Entrées: Un ensemble de données monotones $\mathcal{M}$ et un reject-set R^+ représentant $\lambda^+_{\mathcal{M}}$
Sorties: Un ensemble de règles compatibles avec $\mathcal{M}$

fonction SIMPLIFICATION$^\wedge(\mathcal{M}, R^+)$
  $R \leftarrow \{\}$
  pour chaque $r \in R^+$ **faire**
    $I \leftarrow \{\}$
    pour $i \in [n]$ par ordre croissant de $u^\wedge(\alpha_i, i)$ **faire**
      si $r \backslash (I \cup \{i\})$ est compatible avec $\mathcal{M}$ **alors**
        $I \leftarrow I \cup \{i\}$
    $R \leftarrow R \cup \{r \backslash I\}$
  retourner R^*

Enfin, les algorithmes d'apprentissage SRL$^\vee$ et SRL$^\wedge$ sont définies de la manière suivante.

SRL$^\vee(\mathcal{D})$. *On considère un ensemble d'observations $\mathcal{D}$.*

1. *Effectuer un relabeling de $\mathcal{D}$ qui minimise la MAE. On appelle $\mathcal{M}$ l'ensemble relabelisé.*
2. *Représenter $\lambda^-_{\mathcal{M}}$ par un ensemble de règles R^- :*
$$R^- \leftarrow \left\{\mathbf{x} \nearrow y \;\middle|\; (\mathbf{x}, y) \in \mathcal{M}\right\}^*.$$
3. *Simplifier les règles de R^- :*
$$R \leftarrow \text{SIMPLIFICATION}^\vee(\mathcal{M}, R^-).$$

Le résultat de la méthode est l'ensemble R.

SRL$^\wedge(\mathcal{D})$. *On considère un ensemble d'observations $\mathcal{D}$.*

1. *Effectuer un relabeling de $\mathcal{D}$ qui minimise la MAE. On appelle $\mathcal{M}$ l'ensemble relabelisé.*
2. *Représenter $\lambda^+_{\mathcal{M}}$ par un ensemble de règles R^+ :*
$$R^+ \leftarrow \left\{\mathbf{x} \searrow y \;\middle|\; (\mathbf{x}, y) \in \mathcal{M}\right\}^*.$$
3. *Simplifier les règles de R^+ :*
$$R \leftarrow \text{SIMPLIFICATION}^\wedge(\mathcal{M}, R^+).$$

Le résultat de la méthode est l'ensemble R.

Exemple 74. Illustrons les étapes 2 et 3 de SRL$^\vee$. Soit $L = \{0, 1\}$ et $X_1 = X_2 = \{0, a, b, 1\}$ tels que $0 < a < b < 1$, et soit

$$\mathcal{M} = \big\{\big((0,1),0\big),\ \big((a,b),0\big),\ \big((a,a),0\big),\ \big((1,b),1\big),\ \big((a,1),1\big)\big\}$$

Après l'étape 2, on obtient l'ensemble $R^- = \{r_1, r_2\}$ avec

$$r_1 = (1,\ b) \nearrow 1 \quad \text{et} \quad r_2 = (a,\ 1) \nearrow 1.$$

Notez que les observations dont la classe est 0 ne produisent que des règles de sélection dont la conclusion est $y \geq 0$, qui n'apportent pas d'information. Voyons maintenant quelles règles sont générées par l'Algorithme 13. On calcule les valeurs de $u_i^\vee$ pour les observations dont la classe est 1. Pour $((1,b),1)$ on a

$$u_1^\vee(1,1) = 4 \quad \text{et} \quad u_2^\vee(b,1) = 3,$$

et pour $((a,1),1)$ on a

$$u_1^\vee(a,1) = 3 \quad \text{et} \quad u_2^\vee(1,1) = 3.$$

Commençons par traiter la règle $(1,b) \nearrow 1$: puisque $u_1^\vee(1,1) > u_2^\vee(b,1)$, on regarde d'abord le 2-ème attribut. Celui-ci peut être abaissé à 0, puisque la règle $(1,0) \nearrow 1$ est compatible avec toutes les observations dans $\mathcal{M}$. Le premier attribut ne peut pas être abaissé à 0, car la règle $(0,0) \nearrow 1$ est incompatible avec toutes les observations dont la classe est 0. On traite ensuite $(a,1) \nearrow 1$. Aucun attribut de cette règle ne peut être abaissé à 0, puisque $(0,1) \nearrow 1$ est incompatible avec $((0,1),0)$, et puisque $(a,0) \nearrow 1$ est incompatible avec $((a,b),0)$ et $((a,a),0)$. Donc la fonction SIMPLIFICATION$^\vee$ retourne $R = \{r_1', r_2\}$ avec

$$r_1' = \big[(1,\ 0) \nearrow 1\big] \quad \text{et} \quad r_2 = \big[(a,\ 1) \nearrow 1\big].$$

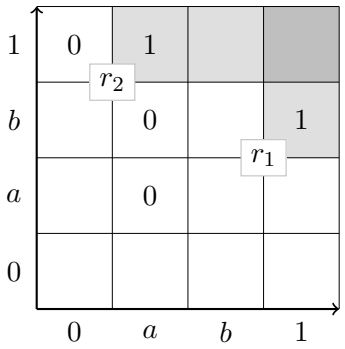


FIGURE 6.2 – Observations de $\mathcal{M}$ et règles de R^-.

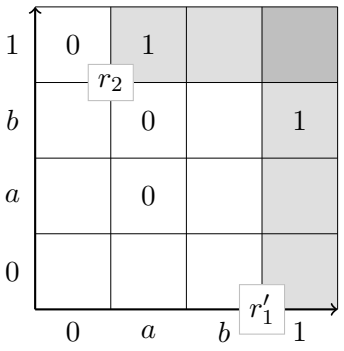


FIGURE 6.3 – Observations de $\mathcal{M}$ et règles issues de SIMPLIFICATION$^\vee(\mathcal{M}, R^-)$.

6.5.2 Évaluation de la méthode

Nous comparons la précision prédictive des deux variantes de SRL avec celle de VC-DomLEM. Nous nous basons sur les jeux de données présentés dans la Table 6.1. Ces jeux de données sont identiques à ceux utilisé dans [19], et nous ont été fournis par Roman Slowinski. Notez que certains de ces jeux de données contiennent des paires d'observations anti-monotones, que nous interprétons comme du bruit (ce bruit est traité par la première étape des deux variantes de SRL).

Id	Nom	Nb. observations	Nb. attributs	Nb. classes	Description/Origine
1	breast-c	286	8	2	Breast-cancer diagnoses[4]
2	breast-w	699	9	2	Breast-cancer diagnoses[5]
3	car	1296	6	4	Car evaluations
4	CPU	209	6	4	CPU performance evaluation
5	bank-g	1411	16	2	Greek banks evaluation
6	fame	1328	10	5	Firm financial evaluation
7	denbosch	119	8	2	House pricing [33]
8	ERA	1000	4	9	Employees rejection/acceptance [8][6]
9	ESL	488	4	9	Employees selection [8][6]
10	LEV	1000	4	5	Lectures evaluations [8][6]
11	SWD	1000	10	4	Social workers decisions [8][6]
12	windsor	546	10	4	House pricing [3]

TABLE 6.1 – Description des jeux de données.

Dans la suite de ce chapitre, on évaluera la précision d'un algorithme d'apprentissage $\mathscr{A}$ sur un jeu de données $\mathcal{D}$ par le test suivant.

Évaluation d'un algorithme d'apprentissage$(\mathscr{A}, \mathcal{D})$**.** *On considère un jeu de données $\mathcal{D}$ et un algorithme d'apprentissage $\mathscr{A}$. Le test consiste en 10 validations croisées en 10 plis. Pour chaque validation croisée, on mesure la performance moyenne de $\mathscr{A}$ (MER et MAE) obtenue sur les 10 plis de $\mathcal{D}$.*

Le test retourne, pour la MAE et pour la MER, la moyenne et l'écart type des résultats des 10 validations croisées.

Nous effectuons ce test pour les quatre algorithmes d'apprentissage suivants (sur chacun des 12 jeux de données de la table) :

- $\text{SRL}^{\vee}$,
- $\text{SRL}^{\wedge}$,
- une méthode d'apprentissage composée des deux premières étapes de $\text{SRL}^{\vee}$ (qui renvoie $\lambda^{-}_{\mathcal{M}}$),
- une méthode d'apprentissage composée des deux premières étapes de $\text{SRL}^{\wedge}$ (qui renvoie $\lambda^{+}_{\mathcal{M}}$).

La Table 6.2 donne les résultats obtenus par nos algorithmes ainsi que ceux obtenus par VC-DomLEM (les résultats sont reportés de [19]).

4. `https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)`
5. `https://archive.ics.uci.edu/ml/datasets/breast+cancer`
6. `https://www.cs.waikato.ac.nz/ml/weka/datasets.html`

	$\lambda^{-}_{\mathcal{M}}$		$\lambda^{+}_{\mathcal{M}}$		SRL$^{\vee}$		SRL$^{\wedge}$		VC-DomLEM	
	MER	MAE	MER	MAE	MER	MAE	MER	MAE	MER	MAE
breast-c	0.253 ± 0.010	0.253 ± 0.010	0.276 ± 0.011	0.276 ± 0.011	0.256 ± 0.011	0.256 ± 0.011	0.278 ± 0.007	0.278 ± 0.007	**0.233** ± 0.003	**0.232** ± 0.003
breast-w	0.12 ± 0.006	0.12 ± 0.006	0.055 ± 0.002	0.055 ± 0.002	0.042 ± 0.003	0.042 ± 0.003	0.04 ± 0.003	0.04 ± 0.003	**0.037** ± 0.002	**0.037** ± 0.002
car	0.038 ± 0.002	0.045 ± 0.002	0.04 ± 0.002	0.045 ± 0.002	**0.023** ± 0.001	**0.026** ± 0.002	0.034 ± 0.003	0.038 ± 0.003	0.028 ± 0.001	0.034 ± 0.001
CPU	0.209 ± 0.008	0.225 ± 0.012	0.22 ± 0.004	0.246 ± 0.005	**0.064** ± 0.008	**0.067** ± 0.008	0.099 ± 0.008	0.102 ± 0.011	0.083 ± 0.014	0.083 ± 0.015
bank-g	0.141 ± 0.001	0.141 ± 0.001	0.311 ± 0.006	0.311 ± 0.006	0.083 ± 0.003	0.083 ± 0.003	0.059 ± 0.001	0.059 ± 0.001	**0.046** ± 0.001	**0.046** ± 0.001
fame	0.547 ± 0.004	0.682 ± 0.004	0.61 ± 0.005	0.803 ± 0.007	0.344 ± 0.004	0.367 ± 0.003	0.343 ± 0.005	0.367 ± 0.005	**0.334** ± 0.005	**0.341** ± 0.002
denbosch	0.306 ± 0.018	0.306 ± 0.018	0.23 ± 0.014	0.23 ± 0.014	0.168 ± 0.013	0.168 ± 0.013	0.146 ± 0.008	0.146 ± 0.008	**0.123** ± 0.010	**0.123** ± 0.010
ERA	**0.735** ± 0.010	**1.26** ± 0.011	0.766 ± 0.007	1.3 ± 0.014	**0.73** ± 0.006	**1.251** ± 0.011	0.76 ± 0.002	1.295 ± 0.009	**0.731** ± 0.004	1.307 ± 0.002
ESL	**0.326** ± 0.012	**0.348** ± 0.012	0.344 ± 0.014	0.388 ± 0.014	**0.33** ± 0.010	**0.355** ± 0.013	0.34 ± 0.006	**0.36** ± 0.007	**0.333** ± 0.013	0.37 ± 0.014
LEV	0.371 ± 0.009	0.402 ± 0.007	0.384 ± 0.005	0.42 ± 0.006	**0.364** ± 0.004	**0.394** ± 0.004	0.384 ± 0.004	0.418 ± 0.004	0.444 ± 0.004	0.481 ± 0.004
SWD	**0.425** ± 0.008	**0.443** ± 0.009	**0.419** ± 0.005	**0.443** ± 0.006	**0.418** ± 0.008	**0.435** ± 0.009	**0.422** ± 0.005	0.445 ± 0.006	0.436 ± 0.005	0.454 ± 0.004
windsor	0.513 ± 0.011	0.589 ± 0.011	0.486 ± 0.011	0.587 ± 0.013	0.486 ± 0.014	0.551 ± 0.018	0.472 ± 0.009	0.526 ± 0.010	**0.454** ± 0.008	**0.502** ± 0.006
moyenne	0.332	0.401	0.345	0.425	0.276	0.333	0.281	0.339	0.274	0.334

TABLE 6.2 – Comparaison des erreurs obtenues par l'algorithme SRL avec des ensembles de règles non simplifiées et avec VC-DomLEM. Pour chaque ligne, les scores situés à un écart type ou moins du score le plus bas sont dénotés en gras.

Les résultats montrent que l'étape de simplification des règles de SRL a un effet positif important sur la précision du modèle.

Parmi les méthodes SRL$^{\wedge}$, SRL$^{\vee}$ et VC-DomLEM, aucune ne domine les autres sur l'ensemble des jeux de données. Cependant, SRL$^{\wedge}$ n'obtient de meilleurs résultats que VC-DomLEM que sur SWD. Enfin, les scores de SRL$^{\wedge}$ et de SRL$^{\vee}$, bien que proches en moyenne, différent fortement sur certains jeux de données.

Il est intéressant de constater que la méthode SRL$^{\vee}$, qui est basée sur deux principes simples (ne commettre aucune erreur sur les données d'apprentissage relabélisées et minimiser la taille des règles par une approche gloutonne), obtient des résultats en moyenne proches de ceux de VC-DomLEM, qui fait appel à plusieurs hyper-paramètres et s'appuie sur la théorie bien plus élaborée de la DRSA.

Les distributions moyennes des longueurs des règles (nombre d'attribut actifs) retournées par SRL$^{\vee}$ et SRL$^{\wedge}$ sur l'ensemble des tests sont données, respectivement, dans la Table 6.3 et dans la Table C.1 en annexe.

Dataset	Longueur des règles															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		9	45	46	1											
2	25	65	10	1												
3	18	21	24	23	12	3										
4	48	44	8													
5	6	62	27	5	1											
6	3	29	51	13	3	1										
7	26	64	10													
8	28	67	5													
9	13	35	30	22												
10	14	40	46	1												
11	16	25	37	19	3											
12	3	24	44	26	2	1										

TABLE 6.3 – Distributions des longueurs de règles obtenues par $\text{SRL}^{\vee}$. Chaque ligne donne la distribution de la taille des règles (en pourcentage, arrondi à l'entier supérieur) obtenue sur un jeu de données. Le nombre d'attributs de chaque jeu de données est indiqué par une double-barre verticale.

Ces distributions peuvent être comparées à celles obtenues dans [11] ; les règles générées par $\text{SRL}^{\vee}$, $\text{SRL}^{\wedge}$ et VC-DomLEM sont, dans leur grande majorité, d'une longueur comprise entre 1 et 6 attributs. Cependant, la distribution des longueurs de règles entre ces deux bornes diffère selon les méthodes.

6.6 FUS-couverture minimale

Nous cherchons maintenant à estimer le nombre de FUS nécessaires pour décrire une fonction équivalente à un ensemble de règles appris à partir de données empiriques. Nous posons le problème suivant.

Problème de FUS-couverture minimal. *Soit R un select-set (resp. reject-set). Trouver une $\vee$-FUS-couverture ($\wedge$-FUS-couverture) de R de taille minimale.*

En effet, la taille de la plus petite $\vee$-FUS-couverture (resp. $\wedge$-FUS-couverture) d'un select-set (resp. reject-set) R correspond à la taille minimale de tout ensemble de FUS $\mathbf{S}$ vérifiant

$$\bigvee \mathbf{S} = f_R \quad \Big(\text{resp. } \bigwedge \mathbf{S} = f_R\Big).$$

Afin de mieux cerner la difficulté de ce problème, considérez l'exemple suivant.

Exemple 75. Soit $n = 4$, $L = X_1 = X_2 = X_3 = X_4 = \{0, a, b, 1\}$ et un select-set $R = \{r^1, r^2, r^3, r^4\}$ de domaine L^4 et codomaine L, où

$$\begin{aligned} r^1 &= (a, a, 0, 0) \nearrow b, \\ r^2 &= (a, 0, a, 0) \nearrow b, \\ r^3 &= (a, 0, 0, a) \nearrow b, \\ r^4 &= (0, b, b, b) \nearrow 1. \end{aligned}$$

On peut vérifier que chaque sous-ensemble de R composé de 3 règles est FUS-représentable. En revanche, R n'est pas FUS-représentable. Pour le voir, calculons $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ et μ tels que $S^{\vee}_{\mu,\varphi} = \mathbb{FUS}^{\vee}(R)$. On obtient les NQ locales suivantes :

x	0	a	b	1
$\varphi_1(x)$	0	b	b	1
$\varphi_2(x)$	0	b	b	1
$\varphi_3(x)$	0	b	b	1
$\varphi_4(x)$	0	b	b	1

ainsi que la fonction μ, dont les ensembles focaux sont $\{1,2\}$, $\{1,3\}$, $\{1,4\}$, $\{2,3,4\}$ avec

$$\mu(\{1,2\}) = \mu(\{1,3\}) = \mu(\{1,4\}) = b \quad \mu(\{2,3,4\}) = 1.$$

Si l'on traduit ensuite $S^{\vee}_{\mu,\varphi}$ en règles on obtient l'ensemble $R' = \{r^1, r^2, r^3, r^4, r^5\}$ où

$$r^5 = (0, a, a, a) \nearrow b.$$

On a donc $\mathbb{FUS}^{\vee}(R) > f_R$. La présence de la règle r^5 dans R' vient du fait que $\mu(\{2,3,4\}) = 1$ et $\varphi_2(a) = \varphi_3(a) = \varphi_4(a) = b$.

Cet exemple illustre le fait qu'il existe des select-sets R dont tous les sous-ensembles de taille $n-1$ sont FUS-représentables, tandis que $\mathbb{FUS}^{\vee}(R) > f_R$. Ce fait ne constitue pas une preuve que le problème de FUS-couverture minimale est NP-difficile ; cependant nous ne connaissons pas d'algorithme qui puisse le résoudre en temps polynomial par rapport à n et au nombre de règles. On fait donc appel à une méthode approximative.

La fonction $\vee$-FUS_INTERPOLATION (Algorithme 15) est une approche gloutonne qui traite un problème plus général que le problème de FUS-couverture d'un select-set. Étant donné un select-set R^- représenté par une liste $\mathtt{l}_{R^-}$ et une fonction λ^+ telle que $f_{R^-} \leq \lambda^+$, $\vee$-FUS_INTERPOLATION($\mathtt{l}_{R^-}, \lambda^+$) renvoie une partition $\mathbf{P}$ de R^- telle que

$$f_{R^-} \leq \bigvee \{\mathbb{FUS}^{\vee}(P) \mid P \in \mathbf{P}\} \leq \lambda^+.$$

Lorsque $\lambda^+ = f_{R^-}$, l'Algorithme retourne donc une $\vee$-FUS-couverture de R^-.

Le principe de cette fonction est de construire itérativement une couverture $\mathbf{P}$ de R^-, respectant la contrainte

$$\bigvee \{\mathbb{FUS}^{\vee}(P) \mid P \in \mathbf{P}\} \leq \lambda^+$$

en cherchant à minimiser sa cardinalité.

Algorithme 15: Recherche gloutonne d'une partition minimale $\mathbf{P}$ de l'ensemble R^- telle que $\bigvee \{\mathbb{FUS}^\vee(P) \mid P \in \mathbf{P}\} \leq \lambda^+$.

Entrées: Une liste $\mathtt{l}_{R^-}$ contenant les éléments du select-set R^- et une fonction λ^+ telle que $f_{R^-} \leq \lambda^+$.
Sorties: Une liste $\mathtt{l}_{\boldsymbol{P}}$ qui représente une partition de R^-.

```
1  fonction ∨-FUS_INTERPOLATION(l_{R⁻}, λ⁺)
2      l_P ← liste vide
3      pour chaque r dans l_{R⁻} faire
4          i ← 1
5          isolé ← vrai
6          tant que isolé et i ≤ taille(l_P) faire
7              P ← l_P[i] ∪ {r}
8              si 𝔽𝕌𝕊^∨(P) ≤ λ⁺ alors
9                  l_P[i] ← P
10                 trier l_P par ordre de cardinalités décroissantes
11                 isolé ← faux
12             i ← i + 1
13         si isolé alors
14             ajouter {r} à l_P
15     retourner l_P
```

La version duale de cette fonction, $\wedge$-FUS_INTERPOLATION, qui a pour paramètre une fonction λ^- et une liste de règles de sélection $\mathtt{l}_{R^+}$ telles que $\lambda^- \leq f_{R^+}$, est donnée dans l'Annexe C.2. On abrège respectivement les noms des fonctions $\vee$-FUS_INTERPOLATION et $\wedge$-FUS_INTERPOLATION par $\vee$-FI et $\wedge$-FI.

Proposition 76. Soit un select-set R^- et une fonction λ^+ tels que $f_{R^-} \leq \lambda^+$. Il existe nécessairement un select-set R tel que

$$f_{R^-} \leq f_R \leq \lambda^+$$

et une liste $\mathtt{l}_R$ contenant les éléments de R telle que l'ensemble de FUS $\mathbf{S}$ défini par

$$\mathbf{S} = \left\{\mathbb{FUS}^\vee(P) \mid P \in \vee\text{-FI}(\mathtt{l}_R, \lambda^+)\right\}$$

est un ensemble de FUS de taille minimale vérifiant $f_{R^-} \leq \bigvee \mathbf{S} \leq \lambda^+$.

On dit qu'un tel select-set R est un *argument optimal* de la fonction $\vee$-FI.

Proposition 77. Soit une fonction λ^- et un reject-set R^+ tels que $\lambda^- \leq f_{R^+}$. Il existe nécessairement un reject-set R tel que

$$\lambda^- \leq f_R \leq f_{R^+}$$

et une liste $\mathtt{l}_R$ contenant les éléments de R telle que l'ensemble de FUS $\mathbf{S}$ défini par

$$\mathbf{S} = \left\{\mathbb{FUS}^\wedge(P) \mid P \in \wedge\text{-FI}(\lambda^-, \mathtt{l}_R)\right\}$$

est un ensemble de FUS de taille minimale vérifiant $\lambda^- \leq \bigwedge \mathbf{S} \leq f_{R^+}$.

On dit qu'un tel reject-set R est un *argument optimal* de la fonction $\wedge$-FI.

La preuve de la Proposition 76 est donnée dans l'Annexe B.3. Lorsque $f_{R^-} = \lambda^+$ (resp. $f_{R^+} = \lambda^-$), l'argument optimal est nécessairement R^- (resp. R^+). On a donc les corollaires suivant.

Corollaire 78. Soit un select-set R. Il existe nécessairement une liste $\mathtt{l}_R$ contenant tous les éléments de R, telle que $\bigvee\{\mathbb{FUS}^{\vee}(P) \mid P \in \vee\text{-FI}(\mathtt{l}_R, f_R)\}$ est une FUS couverture minimale de R.

Corollaire 79. Soit un reject-set R. Il existe nécessairement une liste $\mathtt{l}_R$ contenant tous les éléments de R, telle que $\bigwedge\{\mathbb{FUS}^{\wedge}(P) \mid P \in \wedge\text{-FI}(f_R, \mathtt{l}_R)\}$ est une FUS couverture minimale de R.

En revanche, si $f_{R^-} \neq \lambda^+$ (resp. $f_{R^+} \neq \lambda^-$), l'ensemble R^- (resp. R^+) n'est pas nécessairement un argument optimal de $\vee$-FI (resp. $\wedge$-FI). Cela est dû au fait que pour deux select-sets (resp. reject-sets) R et R' tels que $f_R < f_{R'}$, il est possible que $\mathbb{FUS}^{\vee}(R') < \mathbb{FUS}^{\vee}(R)$ (resp. $\mathbb{FUS}^{\wedge}(R') < \mathbb{FUS}^{\wedge}(R)$).

6.6.1 Expérience

Nous utilisons maintenant les fonctions $\vee$-FI et $\wedge$-FI pour évaluer le nombre de FUS nécessaires à la représentation d'ensembles de règles de différentes dimensions. Nous nous basons sur les 12 jeux de données de la Table 6.1, desquels on extrait des règles de décision via SRL$^{\vee}$ et SRL$^{\wedge}$. Pour chacun des ensembles de règles obtenus, nous effectuons le test suivant.

Test de FUS-couverture(R)**.** *Étant donné un select-set R, on répète 1000 fois les étapes suivantes :*

1. *Mélange aléatoire de $\mathtt{l}_R$,*
2. *Calcul de $\vee$-FI$(\mathtt{l}_R, f_R)$ (voir Algorithme 15).*
3. *Calcul de $\wedge$-FI$(f_R, \mathtt{l}_R)$.*

Le résultat du test est composé de la moyenne, du minimum et de l'écart type de $|\vee\text{-FI}(\mathtt{l}_R, f_R)|$ (ainsi que de $|\wedge\text{-FI}(f_R, \mathtt{l}_R)|$), obtenus sur les 1000 répétitions.

Dataset	1	2	3	4	5	6	7	8	9	10	11	12
Taille de $\mathcal{D}$	286	683	1728	209	1411	1328	119	1000	488	1000	1000	546
Taille de R^-	13	38	36	23	178	557	14	25	36	26	38	101
Nb. FUS moyen	6	11	10	6	126	214	8	8	11	9	7	36
écart type	0	0.78	0.75	0.59	0	2.92	0.45	0.73	0.81	0.81	0.85	1.07
min.	6	9	9	6	126	205	8	7	10	8	6	34
Taille de R^+	13	23	38	33	297	618	6	28	46	23	48	83
Nb. FUS moyen	5	12	16	14	149	206	4	9	11	7	11	34
écart type	0.5	0.72	0.97	0.95	0.16	2.83	0	0.76	0.92	0.69	0.82	1.2
min.	5	11	15	12	149	198	4	8	9	6	10	31

TABLE 6.4 – Résultats du test de FUS-couverture.

Notre algorithme n'étant pas optimal, on ne peut pas garantir absolument que les tailles des ensembles de FUS affichées dans la Table 6.4 soient représentatifs des tailles des solutions optimales. Cependant, les Corollaires 78 et 79 ainsi que le fait que les écarts types affichés soient relativement faibles suggèrent qu'elles en sont de bonnes approximations.

6.7 FUS-interpolation

En pratique, on a rarement intérêt à réaliser une traduction exacte d'un ensemble de règles en un ensemble de FUS, puisque les règles elles-mêmes constituent un modèle biaisé des données

à partir desquelles elles sont apprises. Une piste plus pertinente pourrait être d'apprendre directement un ensemble de FUS à partir des données et de comparer la taille de cet ensemble à la taille d'un ensemble de règles appris indépendamment.

Une $\vee$-FUS interpolation d'un ensemble de données $\mathcal{M}$ est un ensemble de FUS $\mathbf{S}$ qui vérifie

$$\lambda^-_{\mathcal{M}} \leq \bigvee \mathbf{S} \leq \lambda^+_{\mathcal{M}}.$$

Une $\wedge$-FUS interpolation d'un ensemble de données $\mathcal{M}$ est un ensemble de FUS $\mathbf{S}$ qui vérifie

$$\lambda^-_{\mathcal{M}} \leq \bigwedge \mathbf{S} \leq \lambda^+_{\mathcal{M}}.$$

Afin d'estimer le nombre de FUS nécessaires à l'interpolation de données empiriques relabélisées, on commence par considérer les problèmes suivants.

$\vee$-FUS interpolation minimale. *Soit un ensemble de données monotones $\mathcal{M}$. Retourner une $\vee$-FUS interpolation de $\mathcal{M}$ de cardinalité minimale.*

$\wedge$-FUS interpolation minimale. *Soit un ensemble de données monotones $\mathcal{M}$. Retourner une $\wedge$-FUS interpolation de $\mathcal{M}$ de cardinalité minimale.*

On se base à nouveau sur les fonctions $\vee$-FI et $\wedge$-FI. Soit un select-set R^- et un reject-set R^+ tels que $f_{R^-} = \lambda^-_{\mathcal{M}}$ et $f_{R^+} = \lambda^+_{\mathcal{M}}$. Des solutions (pas nécessairement optimales) des problèmes de $\vee$-FUS et $\wedge$-FUS interpolation sont donnés respectivement par $\vee$-FI($\mathtt{l}_{R^-}, \lambda^+_{\mathcal{M}}$) et $\wedge$-FI($\lambda^-_{\mathcal{M}}, \mathtt{l}_{R^+}$).

6.7.1 Expérience 1

Nous réalisons une première expérience, basée sur les jeux de données de la section précédente. Cette fois, pour chaque jeu de données, nous réalisons le test suivant.

Test de FUS interpolation($\mathcal{D}$)**.**

1. *On effectue un relabeling de $\mathcal{D}$ qui minimise la MAE. On appelle $\mathcal{M}$ l'ensemble ainsi obtenu.*
2. *On créé deux listes $\mathtt{l}_{R^-}$ et $\mathtt{l}_{R^+}$ représentant le select-set R^- et le reject-set R^+ tels que $f_{R^-} = \lambda^-_{\mathcal{M}}$ et $f_{R^+} = \lambda^+_{\mathcal{M}}$.*
3. *On répète 1000 fois les étapes suivantes :*
   (a) *Mélange aléatoire de $\mathtt{l}_{R^-}$ et $\mathtt{l}_{R^+}$.*
   (b) *Calcul de $\vee$-FI($\mathtt{l}_{R^-}, \lambda^+_{\mathcal{M}}$) (voir Algorithme 15).*
   (c) *Calcul de $\wedge$-FI($\lambda^-_{\mathcal{M}}, \mathtt{l}_{R^+}$).*

Le résultat du test est composé de la taille de $\mathtt{l}_{R^-}$, ainsi que de la moyenne, du minimum et de l'écart type de $|\vee\text{-FI}(\mathtt{l}_{R^-}, \lambda^+_{\mathcal{M}})|$ et de $|\wedge\text{-FI}(\lambda^-_{\mathcal{M}}, \mathtt{l}_{R^+})|$, obtenus sur les 1000 répétitions.

Dataset	1	2	3	4	5	6	7	8	9	10	11	12
Taille de R^-	17	71	36	50	456	811	35	37	46	49	74	128
Nb. FUS moyen	7	9	8	6	19	46	8	3	9	6	5	18
écart type	0.42	1.09	0.64	0.71	1.39	1.73	0.67	0.53	0.69	0.59	0.63	1.03
min.	7	7	8	5	15	42	8	3	8	5	4	15
Taille de R^+	16	54	40	56	610	818	35	41	54	38	61	149
Nb. FUS moyen	6	5	16	3	17	50	4	5	11	8	5	20
écart type	0.31	0.61	0.82	0.54	1.24	2.02	0.5	0.59	0.85	0.68	0.83	1.32
min.	6	4	15	3	14	44	4	4	9	7	4	16

TABLE 6.5 – Résultats du premier test de $\vee$-FUS et $\wedge$-FUS interpolation pour chaque jeu de données.

Dans la section précédente, nous nous sommes basés sur les corollaires des Propositions 76 et 77 pour affirmer que les résultats du test de FUS-couverture constituaient des approximations plausibles du plus petit nombre de FUS nécessaires afin de représenter exactement les ensembles de règles issus des 12 jeux de données.

Afin de faire une telle affirmation dans le cas de la FUS-interpolation, qui est un problème plus général, il nous faudrait connaitre un argument optimal de la fonction $\vee$-FI (resp. $\wedge$-FI) (voir Propositions 76 et 77).

Par conséquent, les résultats de la Table 6.5 doivent être vus comme des bornes supérieures du nombre de FUS nécessaires à l'interpolation de chaque jeu de données, plutôt que comme des estimations fiables.

6.7.2 Expérience 2

Nous ne connaissons pas d'algorithme retournant un argument optimal de $\vee$-FI (resp. $\wedge$-FI) en temps raisonnable.

Cependant, notez que les règles de R^- (resp. R^+) possèdent un grand nombre d'attributs actifs. Il suit donc de la définition de $\mathbb{FUS}^\vee$ (resp. $\mathbb{FUS}^\wedge$) que les ensembles focaux des FUS de $\vee$-FI$(\mathtt{l}_{R^-}, \lambda^+_{\mathcal{M}})$ (resp. $\wedge$-FI$(\lambda^-_{\mathcal{M}}, \mathtt{l}_{R^+})$) sont majoritairement des ensembles de grande cardinalité. Construire des FUS dont les ensembles focaux sont de tailles plus faibles pourrait être un moyen de diminuer le nombre de FUS utilisées dans l'interpolation des données. On peut également faire l'hypothèse que de telles FUS seraient plus interprétables.

Afin d'obtenir des ensembles focaux de plus petites tailles, nous effectuons un test similaire à celui de la sous-section précédente, où R^- (resp. R^+) est remplacé par l'ensemble de règles issu de SRL$^\vee$ (resp. SRL$^\wedge$).

Test de FUS-interpolation 2$(\mathcal{D})$**.**

1. *On effectue un relabeling de $\mathcal{D}$ qui minimise la MAE. On appelle $\mathcal{M}$ l'ensemble ainsi obtenu.*
2. $R^\vee \leftarrow SRL^\vee(\mathcal{M})$ *; création d'une liste* $\mathtt{l}_{R^\vee}$ *contenant les éléments de* $R^\vee$*.*
3. $R^\wedge \leftarrow SRL^\wedge(\mathcal{M})$ *; création d'une liste* $\mathtt{l}_{R^\wedge}$ *contenant les éléments de* $R^\wedge$*.*
4. *On répète 1000 fois les étapes suivantes :*
   (a) *Mélange aléatoire de* $\mathtt{l}_{R^\vee}$ *et de* $\mathtt{l}_{R^\wedge}$*.*
   (b) *Calcul de* $\vee$-FI$(\mathtt{l}_{R^\vee}, \lambda^+_{\mathcal{M}})$ *(voir Algorithme 15).*
   (c) *Calcul de* $\wedge$-FI$(\lambda^-_{\mathcal{M}}, \mathtt{l}_{R^\wedge})$*.*

Le résultat du test est composé des tailles de $R^\vee$ et $R^\wedge$ ainsi que de la moyenne, du minimum et de l'écart type de $|\vee\text{-FI}(\mathtt{l}_{R^\vee}, \lambda^+_{\mathcal{M}})|$ *et de* $|\wedge\text{-FI}(\lambda^-_{\mathcal{M}}, \mathtt{l}_{R^\wedge})|$*, obtenus sur les 1000 répétitions.*

Dataset	1	2	3	4	5	6	7	8	9	10	11	12
Taille de $\mathcal{D}$	286	683	1728	209	1411	1328	119	1000	488	1000	1000	546
Taille de $R^{\vee}$	13	26	36	21	134	493	15	26	44	25	36	96
Nb. FUS moyen	3	4	10	4	16	60	6	5	11	5	4	17
écart type	0.51	0.35	0.55	0.6	0.88	1.98	0	0.57	0.58	0.52	0.48	0.95
min.	3	4	10	4	13	54	6	5	11	4	4	14
Taille de $R^{\wedge}$	14	15	38	25	104	532	8	25	47	27	42	69
Nb. FUS moyen	4	4	15	8	16	70	4	4	10	6	4	17
écart type	0	0.6	0.56	0.48	0.99	2.13	0	0.55	0.8	0.56	0.53	0.9
min.	4	4	15	8	15	64	4	4	9	6	3	16

TABLE 6.6 – Résultats du second test de $\vee$-FUS et $\wedge$-FUS interpolation pour chaque jeu de données.

On constate que les modifications effectuées lors du second test de FUS-interpolation ne diminuent pas le nombre minimal de FUS obtenues sur l'ensemble de jeux de données. Cependant, le choix de l'ensemble de règles issu de SRL a eu une nette influence (positive ou négative selon les cas) sur de nombreux résultats. Il est donc probable que connaitre des arguments optimaux de $\vee$-FI et $\wedge$-FI permettrait de diminuer de manière conséquente le nombre de FUS utilisés par l'interpolation de ces jeux de données.

6.8 Les ensembles de FUS pour la classification monotone

Dans la section précédente nous avons calculé des $\vee$-FUS et $\wedge$-FUS interpolations de données relabelisées. Voyons maintenant quelle est la précision des modèles ainsi obtenus sur de nouvelles observations.

6.8.1 RL-SUF

On définit les algorithmes d'apprentissage suivants.

RL-SUF$^{\vee}(\mathcal{D})$. *On considère un ensemble d'observations $\mathcal{D}$.*

1. *Effectuer un relabeling de $\mathcal{D}$ qui minimise la MAE. On appelle $\mathcal{M}$ l'ensemble relabelisé.*
2. $R \leftarrow \mathrm{SRL}^{\vee}(\mathcal{M})$.
3. *Retourner* $\vee$-FI$(\mathtt{1}_R, \lambda^+_{\mathcal{M}})$.

RL-SUF$^{\wedge}(\mathcal{D})$. *On considère un ensemble d'observations $\mathcal{D}$.*

1. *Effectuer un relabeling de $\mathcal{D}$ qui minimise la MAE. On appelle $\mathcal{M}$ l'ensemble relabelisé.*
2. $R \leftarrow \mathrm{SRL}^{\wedge}(\mathcal{M})$.
3. *Retourner* $\wedge$-FI$(\lambda^-_{\mathcal{M}}, \mathtt{1}_R)$.

Ces algorithmes consistent en la recherche d'un ensemble de FUS $\mathbf{S}$ de taille minimale tel que

$$\lambda^-_{\mathcal{M}} \leq \bigvee \mathbf{S} \leq \lambda^+_{\mathcal{M}}.$$

Autrement dit, l'apprentissage du classifieur est guidé par la minimisation du nombre de FUS dans le modèle (sous la contrainte d'appartenir à l'intervalle $[\lambda^-_{\mathcal{M}}, \lambda^+_{\mathcal{M}}]$).

6.8.2 Évaluation

Nous comparons les performances de RL-SUF$^\vee$ et RL-SUF$^\wedge$ avec celles de SRL$^\vee$ et SRL$^\wedge$. Nous évaluons chacune de ces méthodes selon la procédure de test décrite dans la Sous-section 6.5.2, sur les jeux de données de la Table 6.1. La Table 6.7 reporte le résultat du test pour chaque méthode et chaque jeu de données. Les résultats sont à nouveau comparés à ceux obtenus par VC-DomLEM.

	SRL$^\vee$		SRL$^\wedge$		RL-SUF$^\vee$		RL-SUF$^\wedge$		VC-DomLEM	
	MER	MAE	MER	MAE	MER	MAE	MER	MAE	MER	MAE
breast-c	0.256 ± 0.011	0.256 ± 0.011	0.278 ± 0.007	0.278 ± 0.007	0.257 ± 0.014	0.257 ± 0.014	0.276 ± 0.013	0.276 ± 0.013	**0.233** ± 0.003	**0.232** ± 0.003
breast-w	0.042 ± 0.003	0.042 ± 0.003	0.04 ± 0.003	0.04 ± 0.003	**0.039** ± 0.003	**0.039** ± 0.003	0.04 ± 0.002	0.04 ± 0.002	**0.037** ± 0.002	**0.037** ± 0.002
car	**0.023** ± 0.001	**0.026** ± 0.002	0.034 ± 0.003	0.038 ± 0.003	**0.022** ± 0.002	**0.026** ± 0.002	**0.024** ± 0.002	0.028 ± 0.002	0.028 ± 0.001	0.034 ± 0.001
CPU	**0.064** ± 0.008	**0.067** ± 0.008	0.099 ± 0.008	0.102 ± 0.011	**0.069** ± 0.009	**0.074** ± 0.009	0.1 ± 0.009	0.106 ± 0.010	0.083 ± 0.014	0.083 ± 0.015
bank-g	0.083 ± 0.003	0.083 ± 0.003	0.059 ± 0.001	0.059 ± 0.001	0.076 ± 0.003	0.076 ± 0.003	0.056 ± 0.003	0.056 ± 0.003	**0.046** ± 0.001	**0.046** ± 0.001
fame	0.344 ± 0.004	0.367 ± 0.003	0.343 ± 0.005	0.367 ± 0.005	**0.328** ± 0.005	0.354 ± 0.004	**0.333** ± 0.006	0.362 ± 0.006	0.334 ± 0.005	**0.341** ± 0.002
denbosch	0.168 ± 0.013	0.168 ± 0.013	0.146 ± 0.008	0.146 ± 0.008	0.163 ± 0.014	0.163 ± 0.014	0.146 ± 0.012	0.146 ± 0.012	**0.123** ± 0.010	**0.123** ± 0.010
ERA	**0.73** ± 0.006	**1.251** ± 0.011	0.76 ± 0.002	1.295 ± 0.009	**0.732** ± 0.006	**1.253** ± 0.011	0.759 ± 0.009	1.295 ± 0.015	**0.731** ± 0.004	1.307 ± 0.002
ESL	**0.33** ± 0.010	**0.355** ± 0.013	0.34 ± 0.006	**0.36** ± 0.007	**0.324** ± 0.012	**0.35** ± 0.012	0.34 ± 0.009	**0.361** ± 0.009	**0.333** ± 0.013	0.37 ± 0.014
LEV	**0.364** ± 0.004	**0.394** ± 0.004	0.384 ± 0.004	0.418 ± 0.004	**0.366** ± 0.005	**0.398** ± 0.006	0.384 ± 0.007	0.419 ± 0.008	0.444 ± 0.004	0.481 ± 0.004
SWD	**0.418** ± 0.008	**0.435** ± 0.009	**0.422** ± 0.005	0.445 ± 0.006	0.427 ± 0.009	0.445 ± 0.010	**0.417** ± 0.007	**0.441** ± 0.008	0.436 ± 0.005	0.454 ± 0.004
windsor	0.486 ± 0.014	0.551 ± 0.018	0.472 ± 0.009	0.526 ± 0.010	0.472 ± 0.006	0.534 ± 0.009	0.47 ± 0.012	0.519 ± 0.010	**0.454** ± 0.008	**0.502** ± 0.006
moyenne	0.276	0.333	0.281	0.339	0.273	0.331	0.279	0.337	0.274	0.334

TABLE 6.7 – Comparaison des erreurs obtenues par RL-SUF et par VC-DomLEM.

À l'issue de ces tests, RL-SUF$^\vee$ obtient des scores similaires à ceux de SRL$^\vee$, et RL-SUF$^\wedge$ obtient des scores similaires à ceux de SRL$^\wedge$. Autrement dit, sur la plupart des jeux de données, l'étape d'apprentissage de l'ensemble de FUS semble affecter peu la précision ; la précision des ensembles de FUS obtenus semble avant tout déterminée par celle des ensembles de règles à partir desquels ils sont appris. Parmi les trois méthodes RL-SUF$^\vee$, RL-SUF$^\wedge$ et VC-DomLEM, aucune ne domine les autres sur l'ensemble des jeux de données.

En utilisant les fonctions $\mathbb{S}$-$\mathbb{SET}$ et $\mathbb{R}$-$\mathbb{SET}$, on peut exprimer les résultats de RL-SUF sous forme d'ensemble de règles. Les distributions moyennes des tailles de règles contenues dans les select-sets (resp. reject-sets) équivalents aux modèles issus de RL-SUF$^\vee$ et de RL-SUF$^\wedge$ sont données par les Tables 6.8 et C.1, respectivement.

Dataset \ Longueur des règles	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	5	10	44	40	1											
2	19	74	7	1												
3	16	19	22	26	14	3										
4	39	48	11			3										
5	7	61	27	5	1											
6	2	31	46	14	3	1				3						
7	22	69	10													
8	20	61	4	15												
9	13	36	27	24												
10	10	37	36	18												
11	14	23	40	18	3					2						
12	3	27	48	18	2	1				1						

TABLE 6.8 – Distributions des tailles de règles traduites à partir du résultat de RL-SUF$^\vee$.

Ces distributions sont similaires à celles obtenues par SRL$^\vee$ et SRL$^\wedge$, respectivement. On remarque cependant que les deux variantes de RL-SUF, pour les jeux de données 6, 11 et 12, génèrent des règles de longueur maximale. La présence de ces règles s'explique, pour RL-SUF$^\vee$, par le fait que dans toute capacité $\mu : 2^{[n]} \to L$, on a $\mu([n]) = 1$. Pour RL-SUF$^\wedge$, elle s'explique par le fait que dans toute anti-capacité $\nu : 2^{[n]} \to L$ on a $\nu([n]) = 0$.

6.8.3 Élagage des modèles

Nous cherchons maintenant à déterminer l'effet d'un élagage des modèles issus de RL-SUF, c'est à dire d'une suppression de certaines FUS dans ces modèles. Soit un ensemble de FUS $\mathbf{S}$ issu de RL-SUF$^\vee$, et l'ensemble d'apprentissage relabélisé $\mathcal{M}$ à partir duquel $\mathbf{S}$ a été appris. La fonction $\bigvee \mathbf{S}$ est une interpolation de $\mathcal{M}$. Supprimer des FUS de $\mathbf{S}$ provoque donc une augmentation de l'erreur empirique de $\mathbf{S}$ sur $\mathcal{M}$. Cette augmentation de l'erreur empirique sera utilisée pour réguler l'élagage. On définit la précision de $\mathbf{S}$ sur $\mathcal{M}$ par rapport au taux de mauvaises classifications MER, par :

$$\text{précision}(\bigvee \mathbf{S}, \mathcal{M}) = 1 - \text{MER}(\bigvee \mathbf{S}, \mathcal{M}),$$

et on considère $\rho \in [0, 1]$, qui représente le ratio de précision minimal que l'on souhaite conserver lors de la suppression d'une FUS.

Nous réalisons l'élagage de $\mathbf{S}$ par la fonction RÉDUCTION_MODÈLE (voir Algorithme 22), qui dépend du paramètre ρ, et qui supprime itérativement des FUS de $\mathbf{S}$ selon une approche gloutonne.

En faisant varier la valeur de ρ, on obtient des élagages qui représentent différents compromis entre la taille de l'ensemble de FUS et sa fidélité aux données d'apprentissage relabélisées. Plus la valeur de ρ est élevée, plus on favorise la précision sur les données d'apprentissage par rapport à la réduction de la taille du modèle. Lorsque $\rho > 1$, toutes les FUS sont conservées.

Algorithme 16: Réduction d'un ensemble de FUS.

Entrées: Un ensemble de FUS $\mathbf{S}$, un ensemble d'observations $\mathcal{M}$, un paramètre $\rho \in [0,1]$.
Sorties: Un sous-ensemble de $\mathbf{S}$.

```
fonction RÉDUCTION^∨(S, M, ρ)
    arrêt ← faux
    tant que arrêt = faux faire
        arrêt ← vrai
        pour S ∈ S faire
            si précision(⋁(S\S), M) ≥ ρ * précision(⋁ S, M) alors
                S ← S\S
                arrêt ← faux
    retourner S
```

6.8.4 Expérience

Pour chaque jeu de données de la table, nous effectuons la procédure de test suivante.

Test : élagage des modèles issus de RL-SUF$^\vee(\mathcal{D})$. *On considère un jeu de données $\mathcal{D}$. Le test est composé de 10 validations croisées à 10 plis (100 plis au total).*

1. *Pour chaque pli,*
   (a) *on divise $\mathcal{D}$ en $\mathcal{D}_{\text{app}}$ et $\mathcal{D}_{\text{eval}}$,*
   (b) $\boldsymbol{S} \leftarrow$ RL-SUF$^\vee(\mathcal{D})$,
   (c) *pour chaque valeur de ρ dans* $\{0.95, 0.96, \ldots, 1\}$
      i. $\boldsymbol{S'} \leftarrow$ RÉDUCTION$^\vee(\boldsymbol{S}, \mathcal{M}, \rho)$,
      ii. *on mesure le taux d'erreur du modèle (MER) obtenu par $\boldsymbol{S'}$ sur $\mathcal{D}_{\text{eval}}$.*

Le résulat du test est, pour chaque valeur de ρ, le nombre moyen de FUS utilisées par le modèle et la MER moyenne obtenue, pour les 100 plis.

La Figure 6.4 affiche les résultats du test d'élagage.

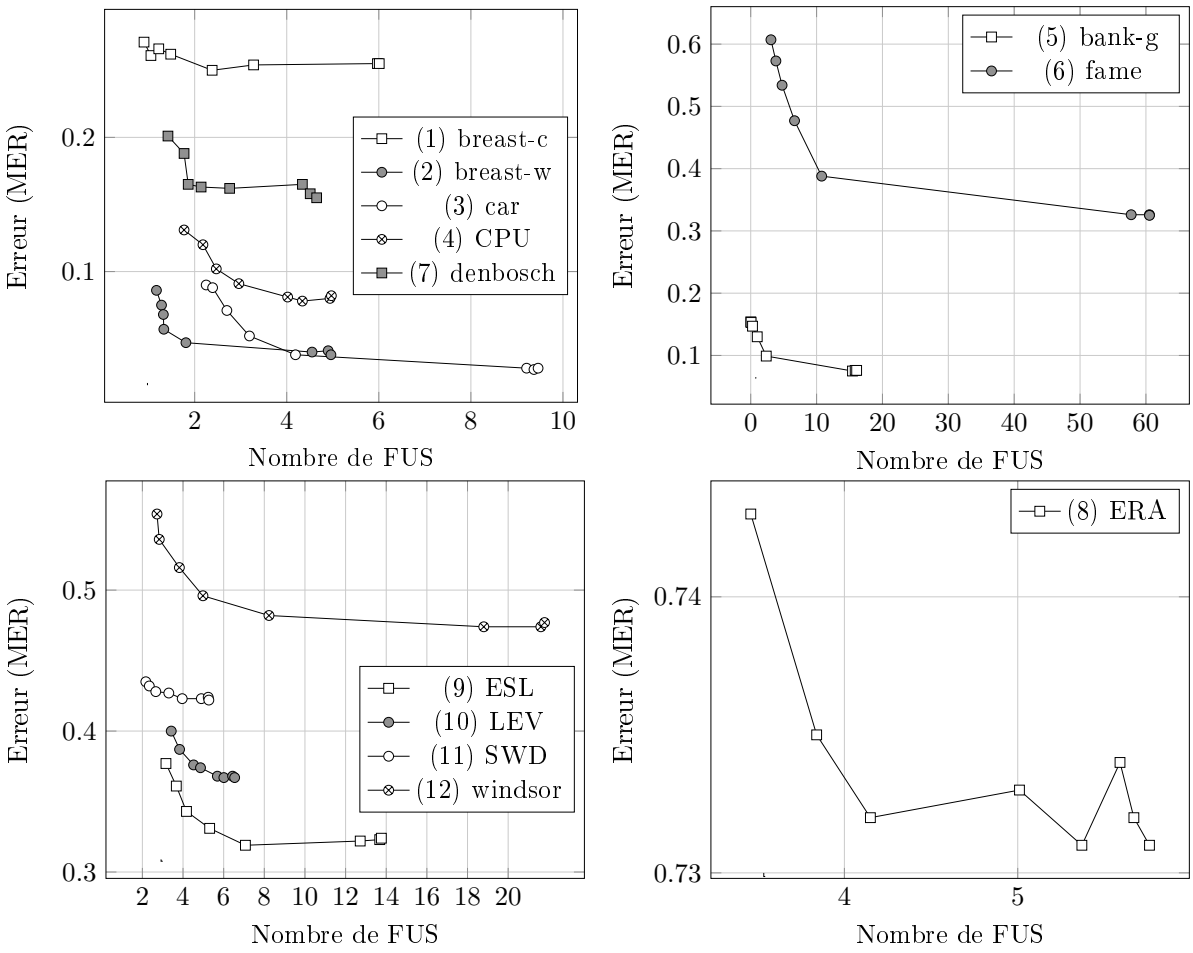


FIGURE 6.4 – Erreur (MER) et nombre de FUS obtenus en moyenne pour chaque test. Chaque courbe correspond à un jeu de données.

Le nombre de FUS nécessaires pour atteindre la précision maximale du modèle varie grandement en fonction des jeux de données. Pour certains jeux de données, on constate que l'élagage permet de réduire sensiblement le nombre de FUS en affectant peu ou pas la précision. C'est le cas pour les jeux de données breast-c, breast-w, ESL et windsor.

Le test d'élagage des modèles issus de RL-SUF$^\wedge$, ainsi que ses résultats, sont présentés dans l'Annexe C.3.

6.9 Conclusion

Dans ce chapitre, nous avons proposé un algorithme d'apprentissage non-paramétrique de règles de décision compétitif avec VC-DomLEM en terme de précision et de tailles des règles. Nous avons ensuite utilisé cet algorithme pour étudier le nombre de FUS nécessaires à la réalisation des tâches suivantes.

- Expression d'un ensemble de règles par un ensemble de FUS équivalent : nous avons donné une estimation empirique du nombre de FUS nécessaires pour représenter des ensembles de règles appris à partir de données empiriques (via SRL). Les résultats obtenus suggèrent que cette approche nécessite l'utilisation d'un trop grand nombre de FUS pour que cela ne

permette d'augmenter l'interprétabilité du modèle.

- Interpolation de données monotones par un ensemble de FUS : nous avons donné des bornes supérieures du nombre de FUS nécessaires à l'interpolation de chacun des jeux de données considérés. Le nombre de FUS utilisées est très variable selon les jeux de données.
- Apprentissage d'un classifieur à base de FUS : les ensembles de FUS interpolant les données d'apprentissage relabelisées, appris via la méthode RL-SUF, obtiennent des performances similaires aux ensembles de règles sur la base desquels ils sont appris. Les méthodes RL-SUF$^\wedge$ et RL-SUF$^\vee$ obtiennent des résultats compétitifs avec ceux de VC-DomLEM.

Afin de réduire la taille des ensembles de FUS appris via RL-SUF, on peut envisager plusieurs améliorations de cette méthode.

Premièrement, il est probable que notre approche du problème de FUS-interpolation puisse être améliorée. Pour cela plusieurs pistes pourraient être explorées :

- La première serait d'effectuer une recherche heuristique des arguments optimaux de $\vee$-FI et $\wedge$-FI. Cette solution permettrait de corriger le principal point faible de notre approche d'interpolation.
- La seconde serait de définir un algorithme calculant une solution de taille minimale au problème de FUS-interpolation. Cependant, il est possible qu'aucun algorithme ne puisse effectuer cette tâche en temps polynomial, d'autant qu'il a été prouvé [27] que le problème d'interpolation par une unique FUS est NP-complet dans le cas où $X_1 = \cdots = X_n$ et où l'on considère la contrainte $\varphi_1 = \cdots = \varphi_n$.
- Enfin, une dernière piste serait de rechercher une solution minimale du problème de FUS-interpolation en utilisant une méthode métaheuristique. Notez que cette piste a déjà été explorée pour l'apprentissage d'intégrales de Sugeno (via des algorithmes génétiques [101] et des essaims particulaires [100]).

Les tests de la méthode d'élagage présentée dans la Sous-section 6.8.3 montrent que sur certains jeux de données, il est possible de réduire la taille des ensembles de FUS issus de RL-SUF en affectant peu leur précision. Ce résultat suggère que la conception d'une méthode d'élagage plus élaborée, ou bien d'un variante de RL-SUF tolérant certaines erreurs sur les données d'apprentissage relabélisées, pourraient également permettre de réduire la taille des ensembles de FUS appris, tout en conservant des précisions similaires.

Conclusion et perspectives

Dans cette thèse nous avons étudié les FLP dans deux directions indépendantes, représentées par deux parties.

Dans la première partie (Chapitres 1 à 4), nous avons contribué à l'étude formelle des FLP définies sur des treillis. Les Chapitres 2 et 3 présentent de nouvelles caractérisations des FLP partielles, tandis que le Chapitre 4 présente une caractérisation des FLP k-maxitives. La contribution principale de cette partie réside dans les méthodes d'interpolation issues de ces résultats. Les caractérisations des Chapitres 2 et 3 débouchent en effet sur deux algorithmes de résolution du problème d'interpolation, qui s'exécutent en temps polynomial par rapport à la taille du treillis considéré, l'arité de la fonction et la taille de son domaine.

Soit une fonction partielle $f : D \to L$ (où $D \subseteq L^n$) sur un treillis L. Dans le cas où L est un treillis fini, la méthode du Chapitre 2 calcule la représentation d'une FLP interpolant f. Nous ne sommes cependant pas capables de calculer les bornes de l'ensemble des FLP interpolant f. La méthode du Chapitre 3 permet de calculer les bornes de l'ensemble des fonctions interpolant f, dans tous les cas où L est un treillis distributif borné qui peut être plongé dans un produit fini de chaînes (cela inclus tous les cas où L est fini).

Les résultats du Chapitre 2 ouvrent de nouvelles possibilités relatives aux algèbres finies. En effet les treillis sont des algèbres possédant deux opérations : $\wedge$ et $\vee$. On appelle généralement *fonction algébrique* (ou *term function* [17]) d'une algèbre A toute fonction pouvant être exprimée à partir de variables, de constantes, et des opérations de l'algèbre A. Les FLP correspondent donc aux fonctions algébriques des treillis. La méthode du Chapitre 2 peut aisément être généralisée à toute algèbre finie possédant des opérations de treillis $\wedge$ et $\vee$, et permet alors de calculer (en temps polynomial) la représentation d'une fonction algébrique interpolant une fonction partielle sur l'algèbre considérée. Des algèbres possédant les opérations de treillis se rencontrent notamment dans le champ des logiques multi-valuées.

Comme nous l'avons vu dans les deux derniers chapitres, l'interpolation peut jouer un rôle dans l'apprentissage d'un modèle de classification monotone. Cependant, il est rare que la tâche de classification monotone soit effectuée dans un contexte où tous les attributs aient le même domaine. De plus l'expressivité des FLP est sévèrement limitée par rapport à celle de l'ensemble des fonctions monotones de L^n à L. Il semble donc que les perspectives d'applications de nos méthodes d'interpolation ne soient pas à chercher dans le domaine de la classification monotone. En revanche, les treillis sont des structures qui apparaissent dans de nombreux domaines. Il se peut donc que le problème d'interpolation apparaisse dans certains de ces domaines.

Enfin, nos méthodes d'interpolation reposent sur l'observation suivante : il existe une FLP interpolant f si et seulement si, pour tout $\mathbf{x}, \mathbf{x'} \in D$, il existe une FLP $p : L^n \to L$ telle que

$$p(\mathbf{x}) = f(\mathbf{x}) \quad \text{et} \quad p(\mathbf{x'}) = f(\mathbf{x'}).$$

Cette équivalence permet de traiter les cas où f n'est pas interpolable par une FLP en terme d'« incompatibilité » de certaines paires $(\mathbf{x}, f(\mathbf{x}))$ et $(\mathbf{x'}, f(\mathbf{x'}))$. Nos méthodes d'interpolation

pourraient donc être étendues afin de modéliser des données bruitées, en nous inspirant des méthodes de restauration de la monotonie dans les jeux de données empiriques (voir Sous-section 5.2.3).

Dans la seconde partie, nous avons étudié l'application des intégrales de Sugeno à la tâche de classification monotone. Pour cela nous avons considéré des généralisations de ces fonctions, appelées FUS. Nous avons défini des fonctions permettant de

- traduire une FUS en un ensemble de règles équivalent,
- traduire un ensemble de règles en une FUS équivalente, lorsqu'une telle FUS existe.

Ces fonctions ont servi de base à une série de tests empiriques ayant pour but d'évaluer le nombre de FUS requis afin de modéliser des données empiriques.

Nous avons proposé une méthode non-paramétrique d'apprentissage de règles de décision (SRL), ainsi qu'une méthode non-paramétrique d'apprentissage d'un ensemble de FUS (RL-SUF). Cette seconde méthode retourne un ensemble de FUS interpolant les données d'apprentissage, après relabélisation. Cet ensemble de FUS est calculé à partir d'un ensemble de règles issu de SRL ; il semble que la précision de l'ensemble de FUS soit en grande partie déterminée par celle de l'ensemble de règles à partir duquel il est appris. Enfin, SRL et RL-SUF obtiennent des résultats compétitifs avec VC-DomLEM sur les jeux de données utilisés dans nos tests.

Les ensembles de FUS sont une alternative aux ensembles de règles de décision. Cependant, il est probable que les ensembles issus de RL-SUF soient de trop grande cardinalité pour être considérés comme avantageux (par rapport aux ensembles de règles issus de SRL, par exemple) sur le plan de l'interprétabilité globale.

Un développement souhaitable des travaux présentés dans le Chapitre 6 serait donc la conception d'algorithmes capables d'apprendre des ensembles de FUS de taille inférieure et de précision au moins similaire à ceux issus de RL-SUF. Des pistes en ce sens sont proposées dans la conclusion du Chapitre 6.

Enfin, dans le but de démontrer un éventuel intérêt des ensembles de FUS en terme d'interprétabilité, il semble pertinent de déterminer de quelle manière les différentes dimensions d'un ensemble de FUS (nombres de FUS, nombre d'ensembles focaux, tailles des ensembles focaux, etc.) et les différentes dimensions d'un ensemble de règles (nombre de règles, tailles des règles, etc.) influencent l'interprétabilité globale de ces deux types de modèles. Pour cela, des études empiriques impliquant des utilisateurs sont certainement nécessaires.

Annexe A

Annexes du Chapitre 2

A.1 Construction d'une représentation de taille minimale

A.1.1 Algorithmes

Algorithme 17: Construction d'une représentation de taille minimale d'une FLP qui coïncide par f sur $\{\mathbf{x}, \mathbf{x'}\}$, si une telle FLP existe (sinon, retourne null).

Entrées: Un treillis L, une fonction partielle $f : D \to L$, deux tuples $\mathbf{x}, \mathbf{x'} \in D$.
Sorties: La représentation d'une FLP.

```
1  fonction SOUS-FLP_MIN(L, x, x', f)
2      l_Я, P, W ← INITIALISATION2(L, x, x')
3      i ← 1
4      tant que i < size(l_Я) faire
5          (a, b) ← l_Я[i]
6          pour j de 0 à i − 1 faire
7              (c, d) ← l_Я[j]
8              MAJ2^∧(l_Я, P, W, (a, b), (c, d))
9              MAJ2^∨(l_Я, P, W, (a, b), (c, d))
10         i ← i + 1
11     retourner P[f(x), f(x')]
```

Algorithme 18: Initialisation de la liste $\mathtt{l}_{\mathcal{Я}}$ et des dictionnaires P et W.

Entrées: Un treillis L, deux tuples $\mathbf{x}, \mathbf{x'} \in L^n$.
Sorties: Une liste $\mathtt{l}_{\mathcal{Я}}$ de couples, un dictionnaire P qui associe une FLP à chacun des couples de $\mathtt{l}_{\mathcal{Я}}$, un dictionnaire W qui associe $\omega(\mathtt{P}[a,b])$ à chaque couple (a,b) de $\mathtt{l}_{\mathcal{Я}}$.

fonction INITIALISATION2($L, \mathbf{x}, \mathbf{x'}$)
  $\mathtt{l}_{\mathcal{Я}} \leftarrow$ liste vide
  $\forall a, b \in L : \mathtt{P}[a,b] \leftarrow$ null
  $\forall a, b \in L : \mathtt{W}[a,b] \leftarrow +\infty$
  pour $i \in [n]$ **faire**
    si $\mathtt{P}[x_i, x'_i] =$ null **alors**
      $\mathtt{P}[x_i, x'_i] \leftarrow$ « var i »
      $\mathtt{W}[x_i, x'_i] \leftarrow 1$
      ajouter (x_i, x'_i) en queue de $\mathtt{l}_{\mathcal{Я}}$
  pour $a \in L$ **faire**
    si $\mathtt{P}[a,a] =$ null **alors**
      $\mathtt{P}[a,a] \leftarrow a$
      $\mathtt{W}[a,a] \leftarrow 1$
      ajouter (a,a) en queue de $\mathtt{l}_{\mathcal{Я}}$
  retourner $\mathtt{l}_{\mathcal{Я}}, \mathtt{P}, \mathtt{W}$

Remarque 80. Si (α, β) n'appartient pas à la liste $\mathtt{l}_{\mathcal{Я}}$, l'instruction « supprimer (α, β) dans $\mathtt{l}_{\mathcal{Я}}$ » n'effectue aucune modification.

Algorithme 19: Mise à jour de $\mathtt{l}_{\mathcal{Я}}$, P et W, qui maintient la liste $\mathtt{l}_{\mathcal{Я}}$ triée.

Entrées: La liste $\mathtt{l}_{\mathcal{Я}}$, le dictionnaire P et W et deux couples $(a,b), (c,d)$.
Sorties: -

fonction MAJ2$^{\wedge}$($\mathtt{l}_{\mathcal{Я}}, \mathtt{P}, \mathtt{W}, (a,b), (c,d)$)
  $(\alpha, \beta) \leftarrow (a \wedge c, b \wedge d)$
  si $\mathtt{W}[\alpha,\beta] > \mathtt{W}[a,b] + \mathtt{W}[c,d] + 1$ **alors**
    $\mathtt{P}[\alpha,\beta] \leftarrow$ null
    supprimer (α,β) dans $\mathtt{l}_{\mathcal{Я}}$
  si $\mathtt{P}[\alpha,\beta] =$ null **alors**
    $\mathtt{P}[\alpha,\beta] \leftarrow (\mathtt{P}[a,b], \wedge, \mathtt{P}[c,d])$
    $\mathtt{W}[\alpha,\beta] \leftarrow \mathtt{W}[a,b] + \mathtt{W}[c,d] + 1$
    ajouter (α,β) à $\mathtt{l}_{\mathcal{Я}}$ en conservant la liste triée

Algorithme 20: Mise à jour de $\mathtt{l}_{\mathcal{Я}}$, P et W, qui maintient la liste $\mathtt{l}_{\mathcal{Я}}$ triée.

Entrées: La liste $\mathtt{l}_{\mathcal{Я}}$, le dictionnaire P et W et deux couples $(a,b), (c,d)$.
Sorties: -

fonction MAJ2$^{\vee}$($\mathtt{l}_{\mathcal{Я}}, \mathtt{P}, \mathtt{W}, (a,b), (c,d)$)
  $(\alpha, \beta) \leftarrow (a \vee c, b \vee d)$
  si $\mathtt{W}[\alpha,\beta] > \mathtt{W}[a,b] + \mathtt{W}[c,d] + 1$ **alors**
    $\mathtt{P}[\alpha,\beta] \leftarrow$ null
    supprimer (α,β) dans $\mathtt{l}_{\mathcal{Я}}$
  si $\mathtt{P}[\alpha,\beta] =$ null **alors**
    $\mathtt{P}[\alpha,\beta] \leftarrow (\mathtt{P}[a,b], \vee, \mathtt{P}[c,d])$
    $\mathtt{W}[\alpha,\beta] \leftarrow \mathtt{W}[a,b] + \mathtt{W}[c,d] + 1$
    ajouter (α,β) à $\mathtt{l}_{\mathcal{Я}}$ en conservant la liste triée

A.1.2 Complexité

Proposition 81. On peut réaliser une implémentation de la fonction SOUS-FLP_MIN dont le temps d'exécution est $O(|L|^4 \log_2 |L| + n)$.

Démonstration. La seule différence significative en terme de complexité entre cette fonction et la fonction originale SOUS-FLP est issue des fonctions $\text{MAJ}^\wedge$ et $\text{MAJ}^\vee$. Une implémentation de $\mathtt{l}_{\text{Я}}$ utilisant un arbre permet de réaliser les instertions et suppression dans $\mathtt{l}_{\text{Я}}$ en $O(\log_2(\text{taille}(\mathtt{l}_{\text{Я}}))$ (voir par exemple [16]). La taille de $\mathtt{l}_{\text{Я}}$ étant au plus $|L|^2$, les insertions et suppressions dans $\mathtt{l}_{\text{Я}}$ s'effectuent en $O(\log_2 |L|)$. □

Annexe B

Démonstrations des Chapitres 5 et 6

B.1 Proposition 63

Proposition 63. Pour tout select-set (resp. reject-set) R, R^* est le plus petit élément de $\mathrm{Eq}^{\vee}(R)$ (resp. $\mathrm{Eq}^{\wedge}(R)$).

Démonstration. On traite le cas où R est un select-set (le cas où R est un reject set peut être traité de manière duale).

Soit un select-set R. La relation d'implication sur les règles de sélection peut être vue comme une relation d'ordre partielle. De plus on a $f_r \leq f_s$ si et seulement si $s \Rightarrow r$. L'ensemble R^* contient l'ensemble des éléments maximaux de R ; par conséquent, pour tout $r \in R$ il existe $s \in R^*$ tel que $f_r \leq f_s$, et donc on a bien

$$f_{R^*} = \bigvee_{r \in R^*} f_r = \bigvee_{r \in R} f_r = f_R.$$

On montre maintenant par contradiction que R^* est inclus dans tout $R' \in \mathrm{Eq}^{\vee}(R)$. Supposons qu'il existe $R' \in \mathrm{Eq}^{\vee}(R)$ tel que $R^* \not\subseteq R'$. Il existe alors une règle $\boldsymbol{\alpha}^1 \nearrow \delta$ qui appartient à R^* et pas à R'. Par définition, R^* ne contient aucune règle $\boldsymbol{\alpha}^1 \nearrow \delta'$, telle que $\delta' > \delta$. On a donc

$$f_{R^*}(\boldsymbol{\alpha}^1) = f_{\boldsymbol{\alpha}^1 \nearrow \delta}(\boldsymbol{\alpha}^1) = \delta.$$

Puisque R^* et R' appartiennent à $\mathrm{Eq}^{\vee}(R)$, $f_{R^*} = f_{R'}$, et donc

$$f_{R'}(\boldsymbol{\alpha}^1) = \delta.$$

Par conséquent R' contient un règle $\boldsymbol{\alpha}^2 \nearrow \delta$ telle que $\boldsymbol{\alpha}^2 \leq \boldsymbol{\alpha}^1$ Si $\boldsymbol{\alpha}^2 = \boldsymbol{\alpha}^1$, la supposition que $(\boldsymbol{\alpha}^1 \nearrow \delta) \notin R'$ est contredite. On suppose donc que $\boldsymbol{\alpha}^2 < \boldsymbol{\alpha}^1$. Puisque $f_{R'}$ est non décroissante, $f_{R'}(\boldsymbol{\alpha}^2) \leq f_{R'}(\boldsymbol{\alpha}^1) = \delta$, et puisque R' contient $\boldsymbol{\alpha}^2 \nearrow \delta$, $f_{R'}(\boldsymbol{\alpha}^2) \geq \delta$. On a donc

$$f_{R^*}(\boldsymbol{\alpha}^2) = f_{R'}(\boldsymbol{\alpha}^2) = \delta,$$

ce qui signifie que R^* contient une règle $\boldsymbol{\alpha}^3 \nearrow \delta$, avec $\boldsymbol{\alpha}^3 \leq \boldsymbol{\alpha}^2 < \boldsymbol{\alpha}^1$. Cela contredit la définition de R^*, car $[\boldsymbol{\alpha}^3 \nearrow \delta] \Rightarrow [\boldsymbol{\alpha}^1 \nearrow \delta]$. □

B.2 Proposition 69 et Corollaire 70

Proposition 69. Pour toute FUS S on a

$$\begin{aligned} S &= f_{\mathbb{S}\text{-}\$\mathbb{ET}(S)} = \mathbb{FUS}^{\vee}(\mathbb{S}\text{-}\$\mathbb{ET}(S)) \\ &= f_{\mathbb{R}\text{-}\$\mathbb{ET}(S)} = \mathbb{FUS}^{\wedge}(\mathbb{R}\text{-}\$\mathbb{ET}(S)). \end{aligned}$$

Démonstration. Nous allons démontrer $S = f_{\mathbb{S}\text{-}\$\mathbb{ET}(S)} = \mathbb{FUS}^{\vee}(\mathbb{S}\text{-}\$\mathbb{ET}(S))$ la preuve de $S = f_{\mathbb{R}\text{-}\$\mathbb{ET}(S)} = \mathbb{FUS}^{\wedge}(\mathbb{R}\text{-}\$\mathbb{ET}(S))$ pouvant être faite de manière analogue. Soit une FUS $S^{\vee}_{\mu,\varphi}$. Nous allons montrer que les assertions suivantes sont toujours vérifiées

A. $S^{\vee}_{\mu,\varphi} \leq \mathbb{FUS}^{\vee}(\mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi}))$ et $S^{\vee}_{\mu,\varphi} \leq f_{\mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi})}$

B. $S^{\vee}_{\mu,\varphi} \geq f_{\mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi})}$

C. $S^{\vee}_{\mu,\varphi} \geq \mathbb{FUS}^{\vee}(\mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi}))$.

A : Soit $\mathbf{x} \in \mathbf{X}$ et $y \in L$ tels que $S^{\vee}_{\mu,\varphi}(\mathbf{x}) = y$. Il existe forcément un ensemble focal $F \in \mathcal{F}(\mu)$ tel que

$$\mu(F) \wedge \bigwedge_{i \in F} \varphi_i(x_i) = y.$$

Par conséquent

$$\mu(F) \geq y \quad \text{and} \quad \forall i \in F,\ \varphi_i(x_i) \geq y.$$

Soit r la règle telle que

$$A^r = F, \quad \delta^r = y \quad \text{et} \quad \forall i \in A^r,\ \alpha^r_i = \bigwedge\{a_i \in X_i \mid \varphi_i(a_i) \geq y\}.$$

notez que pour tout $i \in A^r$ on a $\alpha^r_i \leq x_i$, puisque $\varphi_i(x_i) \geq y$. Par conséquent $f_r(\mathbf{x}) \geq y$. D'après la définition de $\mathbb{S}\text{-}\$\mathbb{ET}$, il existe $s \in \mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi})$ telle que $s \Rightarrow r$. On a donc $f_{\mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi})}(\mathbf{x}) \geq y$.

Soit $S^{\vee}_{\mu',\varphi'}$ la FUS donnée par $S^{\vee}_{\mu',\varphi'} = \mathbb{FUS}^{\vee}(\mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi}))$. D'après la définition de $\mathbb{FUS}^{\vee}$, on a

$$\mu'(A^s) \geq y \quad \text{et} \quad \forall i \in A^s,\ \varphi'_i(x_i) \geq y,$$

et donc $\mathbb{FUS}^{\vee}(\mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi})) \geq y$.

B : Soit $\mathbf{x} \in \mathbf{X}$ et $y \in L$ tels que $f_{\mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi})}(\mathbf{x}) = y$. Il existe forcément une règle $r \in \mathbb{S}\text{-}\$\mathbb{ET}(S^{\vee}_{\mu,\varphi})$ telle que

$$\forall i \in A^r,\ x_i \geq \alpha^r_i \quad \text{et} \quad \delta^r = y.$$

D'après la définition de $\mathbb{S}\text{-}\$\mathbb{ET}$ on a $\mu(A^r) \geq \delta^r$. De plus, pour tout $i \in A^r$,

$$\alpha^r_i = \bigwedge\{a_i \in X_i \mid \varphi_i(a_i) \geq \delta^r\},$$

et donc $\varphi_i(x_i) \geq \varphi_i(\alpha^r_i) \geq \delta^r$. Donc on a

$$\mu(A^r) \wedge \bigwedge_{i \in A^r} \varphi_i(x_i) \quad \geq \quad \delta^r \quad \geq \quad y,$$

et donc $S^{\vee}_{\mu,\varphi}(\mathbf{x}) \geq y$.

C : Soit $S^{\vee}_{\mu',\varphi'}$ la FUS donnée par $\mathbb{FUS}^{\vee}(\mathbb{S}\text{-}\mathbb{SET}(S^{\vee}_{\mu,\varphi}))$. Soit $\mathbf{x} \in \mathbf{X}$ et $y \in L$ tels que $S^{\vee}_{\mu',\varphi'}(\mathbf{x}) = y$. Si $y = 0$ alors $S^{\vee}_{\mu,\varphi}(\mathbf{x}) \geq y$. Supposons donc que $y > 0$.

Il existe nécessairement $F \in \mathcal{F}(\mu')$ tel que

$$\mu'(F) \wedge \bigwedge_{i \in F} \varphi'_i(x_i) = y > 0 \tag{B.1}$$

Par conséquent $\mu'(F) \geq y$ et $\varphi'_i(x_i) \geq y$ pour tout $i \in F$. Puisque $y > 0$, $\mu'(F) > 0$ et donc $F \neq \emptyset$. On commence par montrer que $\mu(F) \geq y$.

- Si $F = [n]$, alors $\mu(F) = 1 \geq y$.
- Si $F \neq [n]$, alors d'après la définition de $\mathbb{FUS}^{\vee}$ on a

  $$\mu'(F) = \bigvee\{\delta^r \mid r \in \mathbb{S}\text{-}\mathbb{SET}(S^{\vee}_{\mu,\varphi}),\ A^r \subseteq F\},$$

  et donc $\mu'(F) \geq y$ implique l'existence d'un règle $r \in \mathbb{S}\text{-}\mathbb{SET}(S^{\vee}_{\mu,\varphi})$ telle que $A^r \subseteq F$ et $\delta^r \geq y$. Par conséquent $\mu(A^r) \geq \delta^r \geq y$.

Pour chaque $i \in F$, on montre que $\varphi_i(x_i) > y$. On a $\varphi'_i(x_i) > 0$ (voir (B.1)) et donc $x_i > 0$.

- Si $x_i = 1$, alors $\varphi_i(x_i) = 1 \geq y$.
- Si $x_i < 1$, alors d'après la définition de $\mathbb{FUS}^{\vee}$ on a

  $$\varphi'_i(x_i) = \bigvee\{\delta^r \mid r \in \mathbb{S}\text{-}\mathbb{SET}(S^{\vee}_{\mu,\varphi}),\ 0 < \alpha^r_i \leq x_i\}$$

  par conséquent $\varphi'(x_i) \geq y$ implique qu'il existe $r \in \mathbb{S}\text{-}\mathbb{SET}(S^{\vee}_{\mu,\varphi})$ telle que $\delta^r = y$ et $0 < \alpha^r_i \leq x_i$. Or d'après la définition de $\mathbb{S}\text{-}\mathbb{SET}$

  $$\alpha^r_i = \bigwedge\{a_i \in X_i \mid \varphi_i(a_i) \geq \delta^r\}.$$

  Par conséquent $\varphi_i(x_i) \geq \varphi_i(\alpha^r_i) \geq \delta^r = y$.

Finalement on a

$$\mu(A^r) \wedge \bigwedge_{i \in A^r} \varphi_i(x_i) \geq y,$$

et donc $S^{\vee}_{\mu,\varphi}(\mathbf{x}) \geq y$. □

Corollaire 70. *Un select-set (resp. reject set) R est FUS-représentable si et seulement si $\mathbb{FUS}^{\vee}(R^*) = f_R$ (resp. $\mathbb{FUS}^{\wedge}(R^*) = f_R$).*

Démonstration. Soit un select-set R. Si R n'est pas FUS-représentable, alors $\mathbb{FUS}^{\vee}(R^*) \neq f_R$. Si R est FUS-représentable, on appelle S la FUS telle que $S = f_R$. D'après la Proposition 69 on a

$$f_{\mathbb{S}\text{-}\mathbb{SET}(S)} = \mathbb{FUS}^{\vee}(\mathbb{S}\text{-}\mathbb{SET}(S)).$$

Or puisque $S = f_R$ on a

$$R^* = \mathbb{S}\text{-}\mathbb{SET}(S)^* = \mathbb{S}\text{-}\mathbb{SET}(S),$$

et donc

$$f_R = f_{R^*} = \mathbb{FUS}^{\vee}(R^*).$$

Dans le cas où R est un reject-set, l'égalité $\mathbb{FUS}^{\wedge}(R^*) = f_R$ peut être prouvée de manière analogue. □

B.3 Proposition 76

Proposition 76. *Soit un select-set R^- et une fonction λ^+ tels que $f_{R^-} \leq \lambda^+$. Il existe nécessairement un select-set R tel que*

$$f_{R^-} \leq f_R \leq \lambda^+$$

et une liste $\mathtt{l}_R$ contenant les éléments de R telle que l'ensemble de FUS $\boldsymbol{S}$ défini par

$$\boldsymbol{S} = \left\{\mathbb{FUS}^{\vee}(P) \mid P \in \vee\text{-FI}(\mathtt{l}_R, \lambda^+)\right\}$$

est un ensemble de FUS de taille minimale FUS vérifiant $f_{R^-} \leq \bigvee \boldsymbol{S} \leq \lambda^+$.

Démonstration. Pour cette démonstration, les listes seront représentés par des tuples.

Soit $\mathbf{S} = \{S_1, \ldots, S_d\}$ un ensemble de FUS de taille minimale tel que $f_{R^-} \leq \mathbf{S} \leq \lambda^+$. Nous allons montrer qu'il existe une liste de règles $\mathtt{l}_R$ telle que $\vee$-FI$(\mathtt{l}_R, \lambda^+)$ retourne un ensemble de taille d et

$$f_{R^-} \leq \left\{\mathbb{FUS}^{\vee}(P) \mid P \in \vee\text{-FI}(\mathtt{l}_R, \lambda^+)\right\} \leq \lambda^+.$$

Supposons qu'il existe une liste $\mathtt{l_P} = (P_1, \ldots, P_d)$ telle que

$$\bigcup_{i=1}^{d} P_i = \bigcup_{i=1}^{d} \mathbb{S}\text{-}\mathbb{SET}(S_i), \tag{B.2}$$

$$\forall i \in [d], \quad \mathbb{FUS}^{\vee}(P_i) \leq \lambda^+, \tag{B.3}$$

$$|P_1| \geq \cdots \geq |P_d|, \tag{B.4}$$

$$\forall i, j \in [d] \text{ tels que } |P_i| \geq |P_j|,\ \forall r \in P_j : \quad \mathbb{FUS}^{\vee}(P_i \cup \{r\}) \not\leq \lambda^+. \tag{B.5}$$

Il suit de (B.2) et (B.3) que

$$f_{R^-} \leq \mathbf{S} \leq \bigvee \left\{\mathbb{FUS}^{\vee}(P_i) \;\middle|\; P_i \in \mathtt{l_P}\right\} \leq \lambda^+.$$

En s'appuyant sur (B.4) et (B.5), on peut définir un liste $\mathtt{l}_R$ telle que $\vee$-FI$(\mathtt{l}_R, \lambda^+) = \mathtt{l_P}$. Pour cela il suffit que $\mathtt{l}_R$:

- contienne tous les éléments de $\bigcup_{i=1}^{d} P_i$,
- soit triée de telle sorte que, pour chaque $i \in \{2, \ldots, d\}$, toutes les règles de P_{i-1} soient positionnées avant les règles de P_i dans $\mathtt{l}_R$.

Afin de démontrer la proposition, il nous reste à montrer qu'il existe bien une liste vérifiant (B.2), (B.3), (B.4) et (B.5). Soit $\mathtt{l_Q}$ la liste définie par

$$\mathtt{l_Q} = (\mathbb{S}\text{-}\mathbb{SET}(S_1), \ldots, \mathbb{S}\text{-}\mathbb{SET}(S_d)).$$

Cette liste vérifie nécessairement (B.2) et (B.3). On définit la fonction g, sur les listes de select-sets de longueur d, par

$$g(R_1, \ldots, R_d) = \begin{cases} (R_1, \ldots, R_d) \text{ si } (R_1, \ldots, R_d) \text{ vérifie (B.5)}, \\ (R'_1, \ldots, R'_d) \text{ sinon}, \end{cases}$$

où $R'_1, \ldots, R'_d$ sont définis de la manière suivante. Premièrement, soient R_i et R_j les plus petit select-sets de $(R_1, \ldots, R_d)$ tels que $|R_i| \geq |R_j|$ et tels que $\exists r \in R_j$, $\mathbb{FUS}^{\vee}(R_i \cup \{r\}) \leq \lambda^+$. Deuxièmement,

$$\forall k \in [d], \quad R'_k = \begin{cases} R_k \cup \{r\} \text{ si } k = i, \\ R_k \backslash \{r\} \text{ si } k = j, \\ R_k \text{ sinon.} \end{cases}$$

On peut aisément vérifier les propriétés suivantes :

- Si $(R_1, \ldots, R_d)$ vérifie (B.2) et (B.3), alors $g(R_1, \ldots, R_d)$ vérifie (B.2) et (B.3).
- Pour toute liste de select-sets $(R_1, \ldots, R_d)$ il existe un entier positif k tel que $g^k(R_1, \ldots, R_d)$ vérifie (B.5).

Par conséquent il existe un entier positif k tel que $g^k(\mathtt{l_Q})$ vérifie (B.2), (B.3), (B.5). Un tri de $g^k(\mathtt{l_Q})$ permet d'obtenir une liste qui vérifie également (B.4). □

Annexe C

Algorithmes et résultats complémentaires du Chapitre 6

C.1 Distributions des longueurs de règles pour $SRL^{\wedge}$

	Longueur des ègles															
Dataset	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	5	10	42	43												
2	7	43	27	18	5											
3			1	23	51	25										
4	5	52	32	12												
5	8	60	25	6	1	1										
6	1	34	42	18	4	1										
7	29	36	32	1	1	1										
8	32	57	11													
9	24	45	24	8												
10	13	39	46	2												
11	3	34	53	10	1											
12	4	8	19	29	22	11	6	1								

TABLE C.1 – Distribution moyenne des tailles des règles des ensembles résultants SRL^

C.2 $\wedge$-FUS_INTERPOLATION

Algorithme 21: Recherche gloutonne d'une partition minimale $\mathbf{P}$ du reject-set R^+ telle que $\bigwedge \{\mathbb{FUS}^{\wedge}(P) \mid P \in \mathbf{P}\} \geq \lambda^-$.

Entrées: Une liste $\mathtt{l}_{R^+}$ contenant les éléments du reject-set R^+ et une fonction λ^- telle que $\lambda^+ \leq f_{R^+}$.
Sorties: Une liste $\mathtt{l}_{\boldsymbol{P}}$ qui représente une partition de R^+.

```
fonction ∨-FUS_INTERPOLATION(λ⁻, l_{R⁺})
    l_P ← liste vide
    pour chaque r dans l_{R⁺} faire
        i ← 1
        isolé ← vrai
        tant que isolé et i ≤ taille(l_P) faire
            P ← l_P[i] ∪ {r}
            si 𝔽𝕌𝕊^∧(P) ≥ λ⁻ alors
                l_P[i] ← P
                trier l_P par ordre de cardinalités décroissantes
                isolé ← faux
            i ← i + 1
        si isolé alors
            ajouter {r} à l_P
    retourner l_P
```

C.3 Élagage des modèles de RL-SUF$^{\wedge}$

Algorithme 22: Réduction d'un ensemble de FUS.

Entrées: Un ensemble de SUF $\mathbf{S}$, un ensemble d'observations $\mathcal{M}$, un paramètre $\rho \in [0, 1]$.
Sorties: Un sous-ensemble de $\mathbf{S}$.

```
fonction RÉDUCTION^∧(S, M, ρ)
    arrêt ← faux
    tant que arrêt = faux faire
        arrêt ← vrai
        pour S ∈ S faire
            si précision(⋀(S\S), M) ≥ ρ * précision(⋀ S, M) alors
                S ← S \ S
                arrêt ← faux
    retourner S
```

Test : élagage des modèles issus de RL-SUF$^\wedge(\mathcal{D})$. *On considère un jeu de données $\mathcal{D}$. Le test est composé de 10 tenfold cross-validations (100 plis au total).*

1. *Pour chaque pli,*
   (a) *on divise $\mathcal{D}$ en $\mathcal{D}_{\text{app}}$ et $\mathcal{D}_{\text{eval}}$,*
   (b) $\boldsymbol{S} \leftarrow \text{RL-SUF}^\wedge(\mathcal{D})$,
   (c) *pour chaque valeur de ρ dans* $\{0.95, 0.96, \ldots, 1\}$
      i. $\boldsymbol{S'} \leftarrow \text{RÉDUCTION}^\wedge(\boldsymbol{S}, \mathcal{M}, \rho)$,
      ii. *on mesure le taux d'erreur du modèle (MER) obtenu par $\boldsymbol{S'}$ sur $\mathcal{D}_{\text{eval}}$.*

Le résulat du test est, pour chaque valeur de ρ, le nombre moyen de FUS utilisés par le modèle et la MER moyenne obtenue, pour les 100 plis.

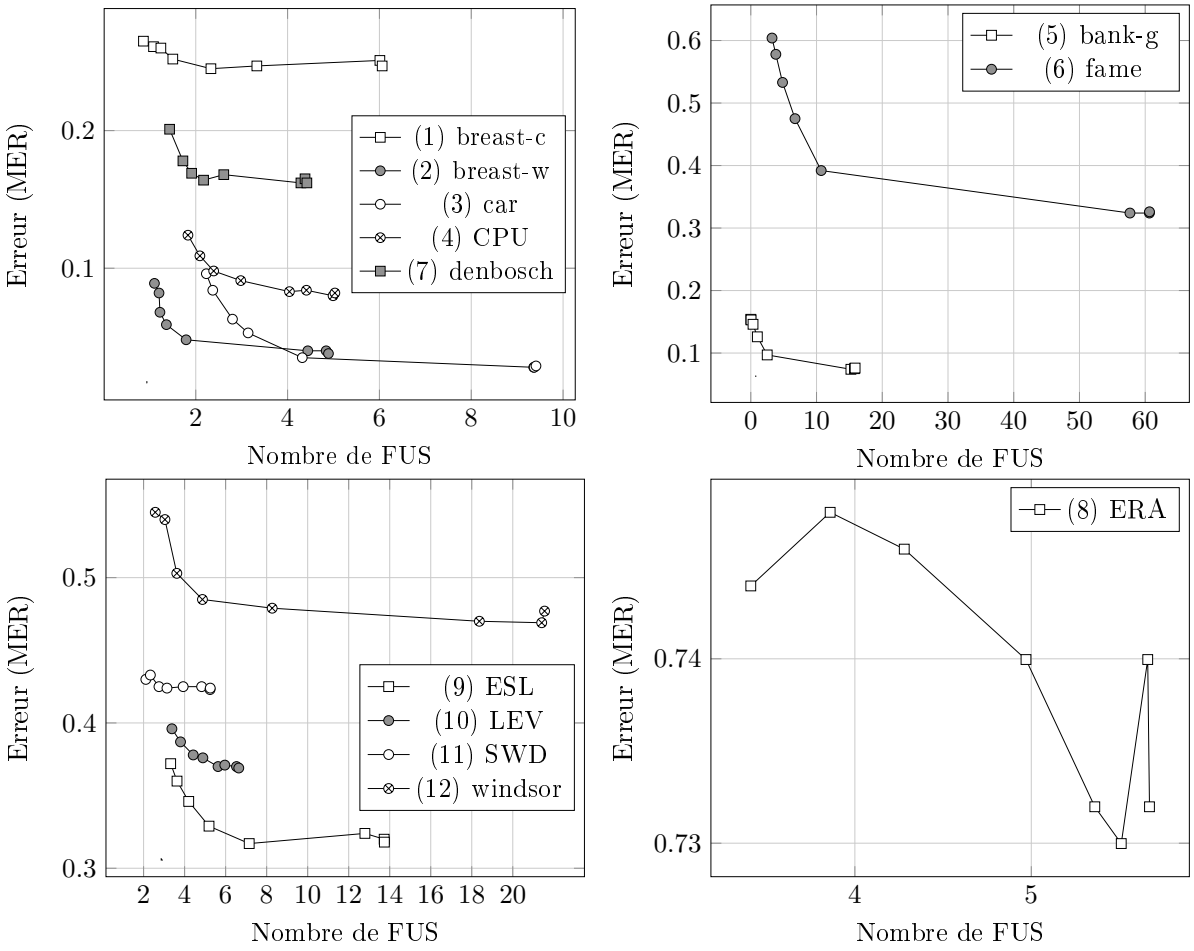


FIGURE C.1 – Résultats du test dual d'apprentissage paramétrique d'un ensemble de FUS. Erreur (MR) et nombre de FUS obtenus en moyenne pour chaque test. Chaque courbe correspond à un jeu de données.

C.4 Distributions des longueurs de règles pour RL-SUF^

Dataset	Longueur des règles															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	9	19	39	33												
2	4	60	23	10	3											
3			1	19	47	34										
4	7	58	20	10		5										
5	8	62	22	6	1	1										
6	1	37	40	14	4	1				3						
7	24	39	35	2	1											
8	26	53	9	12												
9	20	44	20	17												
10	9	34	41	16												
11	2	30	48	10	1					9						
12	3	7	22	28	20	11	5	1		6						

Bibliographie

[1] A. Agresti. *Analysis of ordinal categorical data.* Wiley series in probability and statistics. Wiley, Hoboken, N.J, 2nd ed edition, 2010. OCLC : ocn445480016.

[2] E.E. Altendorf, A.C. Restificar, and T.G. Dietterich. Learning from Sparse Data by Exploiting Monotonicity Constraints. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 18–26, Arlington, Virginia, United States, 2005. AUAI Press.

[3] P.M. Anglin and R. Gençay. Semiparametric estimation of a hedonic price function. *Journal of Applied Econometrics*, 11(6) :633–648, November 1996.

[4] K.A. Baker and A.F. Pixley. Polynomial interpolation and the Chinese Remainder Theorem for algebraic systems. *Mathematische Zeitschrift*, 143(2) :165–174, June 1975.

[5] H-J. Bandelt. Tolerance relations on lattices. *Bulletin of the Australian Mathematical Society*, 23(3) :367–381, 1981.

[6] N. Barile. *Studies in Learning Monotonic Models from Data.* PhD Thesis, Dutch Research School for Information and Knowledge Systems, 2014.

[7] N. Barile and A.J. Feelders. Nonparametric Monotone Classification with MOCA. In *2008 Eighth IEEE International Conference on Data Mining*, pages 731–736, December 2008.

[8] A. Ben-David. Monotonicity Maintenance in Information-Theoretic Machine Learning Algorithms. *Machine Learning*, 19(1) :29–43, April 1995.

[9] A. Ben-David, L. Sterling, and Y-H. Pao. Learning and classification of monotonic ordinal concepts. *Computational Intelligence*, 5(1) :45–49, January 1989.

[10] G. Birkhoff. *Lattice Theory.* American Mathematical Society, New York, USA, 3rd edition, 1967.

[11] J. Blaszczyński, R. Slowiński, and M. Szelag. VC-DomLEM : Rule induction algorithm for variable consistency rough set approaches. Technical report, Research Report RA-07/09, Poznań University of Technology, 2009.

[12] D. Bouyssou, T. Marchant, and M. Pirlot. A Conjoint Measurement Approach to the Discrete Sugeno Integral. In *The Mathematics of Preference, Choice and Order*, Studies in Choice and Welfare, pages 85–109. Springer, Berlin, Heidelberg, 2009.

[13] Q. Brabant and M. Couceiro. k-maxitive Sugeno integrals as aggregation models for ordinal preferences. *Fuzzy Sets and Systems*, 343 :65–75, July 2018.

[14] P. Brass. *Advanced Data Structures.* Cambridge University Press, Leiden, 2008. OCLC : 437234507.

[15] L. Breiman. *Classification and Regression Trees.* Routledge, October 2017.

[16] M.R. Brown and R.E. Tarjan. Design and Analysis of a Data Structure for Representing Sorted Lists. *SIAM Journal on Computing*, 1980.

[17] S.N. Burris and H. Sankappanavar. *A Course in Universal Algebra*, volume 91. 1981.

[18] J. Błaszczyński, S. Greco, R. Slowiński, and M. Szeląg. Monotonic Variable Consistency Rough Set Approaches. *International Journal of Approximate Reasoning*, 50(7) :979–999, July 2009.

[19] J. Błaszczyński, R. Słowiński, and M. Szeląg. Sequential covering rule induction algorithm for variable consistency rough set approaches. *Information Sciences*, 181(5) :987–1002, March 2011.

[20] J.S. Cardoso and R. Sousa. Measuring the Performance of Ordinal Classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(08) :1173–1195, December 2011.

[21] R. Chandrasekaran, Y.U. Ryu, V.S. Jacob, and S. Hong. Isotonic Separation. *INFORMS Journal on Computing*, 17(4) :462–474, November 2005.

[22] C-C. Chen and S-T. Li. Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications*, 41(16) :7235–7247, November 2014.

[23] M. Couceiro, D. Dubois, H. Prade, and A. Rico. Enhancing the Expressive Power of Sugeno Integrals for Qualitative Data Analysis. In *Advances in Fuzzy Logic and Technology 2017*, Advances in Intelligent Systems and Computing, pages 534–547. Springer, Cham, September 2017.

[24] M. Couceiro, D. Dubois, H. Prade, A. Rico, and T. Waldhauser. General Interpolation by Polynomial Functions of Distributive Lattices. In *Advances in Computational Intelligence*, Communications in Computer and Information Science, pages 347–355. Springer, Berlin, Heidelberg, July 2012.

[25] M. Couceiro, D. Dubois, H. Prade, and T. Waldhauser. Decision-Making with sugeno integrals - bridging the gap between multicriteria evaluation and decision under uncertainty. *Order*, 33 :517–535, 2016.

[26] M. Couceiro and J.-L. Marichal. Characterizations of discrete Sugeno integrals as polynomial functions over distributive lattices. *Fuzzy Sets and Systems*, 161(5) :694–707, 2010.

[27] M. Couceiro, M. Maróti, T. Waldhauser, and L. Zadori. Computing version spaces in the qualitative approach to multicriteria decision aid. *À paraître dans : International Journal of Foundations of Computer Science*.

[28] M. Couceiro and T. Waldhauser. Sugeno Utility Functions I : Axiomatizations. In V. Torra, Y. Narukawa, and M. Daumas, editors, *Modeling Decisions for Artificial Intelligence*, Lecture Notes in Computer Science, pages 79–90. Springer Berlin Heidelberg, 2010.

[29] M. Couceiro and T. Waldhauser. Sugeno Utility Functions II : Factorizations. In V. Torra, Y. Narukawa, and M. Daumas, editors, *Modeling Decisions for Artificial Intelligence*, Lecture Notes in Computer Science, pages 91–103. Springer Berlin Heidelberg, 2010.

[30] M. Couceiro and T. Waldhauser. Interpolation by polynomial functions of distributive lattices : a generalization of a theorem of R. L. Goodstein. *Algebra universalis*, 69(3) :287–299, May 2013.

[31] M. Couceiro and T. Waldhauser. Pseudo-polynomial functions over finite distributive lattices. *Fuzzy Sets and Systems*, 239 :21–34, March 2014.

[32] Y. Crama and P.L. Hammer. *Boolean Functions : Theory, Algorithms, and Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011.

[33] H. Daniels and B. Kamp. Application of MLP Networks to Bond Rating and House Pricing. *Neural Computing & Applications*, 8(3) :226–234, August 1999.

[34] H. Daniels and M.V. Velikova. Derivation of Monotone Decision Models from Non-Monotone Data. *Research Group : Operations Research*, 2003.

[35] H. Daniels and M.V. Velikova. Derivation of monotone decision models from noisy data. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(5) :705–710, 2006.

[36] B.A. Davey and H.A. Priestly. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, UK, 2002.

[37] K. Dembczyński, W. Kotłowski., and R. Słowiński. Learning Rule Ensembles for Ordinal Classification with Monotonicity Constraints. *Fundamenta Informaticae*, 94(2) :163–178, 2009.

[38] R.P. Dilworth. A Decomposition Theorem for Partially Ordered Sets. *Annals of Mathematics*, 51(1) :161–166, 1950.

[39] M. Doumpos and C. Zopounidis. Monotonic support vector machines for credit risk rating. *New Mathematics and Natural Computation*, 05(03) :557–570, November 2009.

[40] D. Dubois, C. Durrieu, H. Prade, A. Rico, and Y. Ferro. Extracting Decision Rules from Qualitative Data Using Sugeno Integral : A Case-Study. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 14–24. Springer, 2015.

[41] D. Dubois and H. Fargier. Making Discrete Sugeno Integrals More Discriminant. *International Journal of Approximate Reasoning*, 50(6) :880–898, June 2009.

[42] D. Dubois, J-L. Marichal, H. Prade, M. Roubens, and R. Sabbadin. The use of the discrete sugeno integral in decision-making : a survery. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 09(05) :539–561, October 2001.

[43] W. Duivesteijn and A.J. Feelders. Nearest Neighbour Classification with Monotonicity Constraints. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 301–316. Springer, Berlin, Heidelberg, September 2008.

[44] A.J. Feelders. Monotone Relabeling in Ordinal Classification. In *2010 IEEE International Conference on Data Mining*, pages 803–808, December 2010.

[45] A.J. Feelders and T. Kolkman. Exploiting monotonicity constraints to reduce label noise : An experimental evaluation. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2148–2155, 2016.

[46] A.J. Feelders and L.C. Van Der Gaag. Learning Bayesian network parameters under order constraints. *International Journal of Approximate Reasoning*, 42(1) :37–53, May 2006.

[47] J. Fürnkranz. Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, 13(1) :3–54, February 1999.

[48] J. Fürnkranz, E. Hüllermeier, and S. Vanderlooy. Binary Decomposition Methods for Multipartite Ranking. In W. Buntine, M. Grobelnik, D. Mladenić, and J. Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 359–374. Springer Berlin Heidelberg, 2009.

[49] W.V. Gehrlein. On methods for generating random partial orders. *Operations Research Letters*, 5(6) :285–291, December 1986.

[50] S. González, F. Herrera, and S. García. Monotonic Random Forest with an Ensemble Pruning Mechanism based on the Degree of Monotonicity. *New Generation Computing*, 33(4) :367–388, July 2015.

[51] R.L. Goodstein. The Solution of Equations in a Lattice. *Proceedings of the Royal Society of Edinburgh Section A : Mathematics*, 67(3) :231–242, 1967.

[52] M. Grabisch. The application of fuzzy integrals in multicriteria decision making. *European Journal of Operational Research*, 89(3) :445–456, March 1996.

[53] M. Grabisch. L'utilisation de l'intégrale de Choquet en aide multicritère à la décision. *Newsletter of the European Working Group" Multicriteria Aid for Decisions*, 3(14) :5–10, 2006.

[54] M. Grabisch and C. Labreuche. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 175(1) :247–286, March 2010.

[55] M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap. *Aggregation Functions*, volume 127 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, New York, NY, USA, 2009.

[56] S. Greco, editor. *Multiple Criteria Decision Analysis : State of the Art Surveys*. International Series in Operations Research & Management Science. Springer-Verlag, New York, 2005.

[57] S. Greco, B. Matarazzo, and R. Slowinski. Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research*, 129(1) :1–47, February 2001.

[58] S. Greco, B. Matarazzo, and R. Słowiński. Axiomatic characterization of a general utility function and its particular cases in terms of conjoint measurement and rough-set decision rules. *European Journal of Operational Research*, 158(2) :271–292, October 2004.

[59] G. Grätzer. *General Lattice Theory*. Springer Science & Business Media, Berlin, Germany, 2003.

[60] P.A. Gutiérrez, M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernández-Navarro, and C. Hervás-Martínez. Ordinal Regression Methods : Survey and Experimental Study. *IEEE Transactions on Knowledge and Data Engineering*, 28(1) :127–146, January 2016.

[61] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer-Verlag, New York, 2 edition, 2009.

[62] E.M. Helsper, L.C. Van Der Gaag, A.J. Feelders, W.L.A. Loeffen, P.L. Geenen, and A.R.W. Elbers. Bringing Order into Bayesian-network Construction. In *Proceedings of the 3rd International Conference on Knowledge Capture*, K-CAP '05, pages 121–128, New York, NY, USA, 2005. ACM.

[63] Q. Hu, X. Che, L. Zhang, D. Zhang, M. Guo, and D. Yu. Rank Entropy-Based Decision Trees for Monotonic Classification. *IEEE Transactions on Knowledge and Data Engineering*, 24(11) :2052–2064, November 2012.

[64] S. Ikiz and V.K. Garg. Efficient Incremental Optimal Chain Partition of Distributed Program Traces. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 18–18, 2006.

[65] K. Kaarli and A.F. Pixley. *Polynomial Completeness in Algebraic Systems*. Chapman & Hall/CRC, Boca Raton, USA, 2000.

[66] D. Kagaris and S. Tragoudas. Maximum weighted independent sets on transitive graphs and applications. *Integration, the VLSI Journal*, 27(1) :77–86, 1999.

[67] R.M. Karp. Reducibility among Combinatorial Problems. In R.E. Miller, J.W. Thatcher, and J.D. Bohlinger, editors, *Complexity of Computer Computations : Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, The IBM Research Symposia Series, pages 85–103. Springer US, Boston, MA, 1972.

[68] B. Kim and F. Doshi-Velez. Interpretable Machine Learning : The fuss, the concrete and the questions, 2017.

[69] M. Kindermann. Über die Äquivalenz von Ordnungspolynomvollständigkeit und Toleranzeinfachheit endlicher Verbände. pages 145–149, 1979.

[70] W. Kotlowski. *Statistical Approach to Ordinal Classification with Monotonicity Constraints*. PhD thesis, Poznań University of Technology, 2008.

[71] W. Kotlowski and R. Slowinski. On Nonparametric Ordinal Classification with Monotonicity Constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(11) :2576–2589, 2013.

[72] Ling Li and Hsuan-tien Lin. Ordinal Regression by Extended Binary Classification. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 865–872. MIT Press, 2007.

[73] S. Lievens, B. De Baets, and K. Cao-Van. A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. *Annals of Operations Research*, 163(1) :115–142, October 2008.

[74] J.-L. Marichal. *Aggregation Operators for Multicriteria Decision Aid*. PhD Thesis, Institute of Mathematics, University of Liège, 1998.

[75] J-L. Marichal. On Sugeno integral as an aggregation function. *Fuzzy Sets and Systems*, 114(3) :347–365, 2000.

[76] J-L. Marichal. Weighted lattice polynomials. *Discrete Mathematics*, 309(4) :814–820, March 2009.

[77] J-L. Marichal, P. Meyer, and M. Roubens. Sorting multi-attribute alternatives : The TOMASO method. *Computers & Operations Research*, 32(4) :861–877, April 2005.

[78] R. Mesiar. k-order Pan-discrete fuzzy measures. In *Proceedings of the 7th International Fuzzy Systems Association World Congress (IFSA)*, pages 25–29, Prague, 1997.

[79] R. Mesiar. Generalizations of k-order additive discrete fuzzy measures. *Fuzzy Sets and Systems*, 102(3) :423–428, 1999.

[80] R.H. Möhring. Algorithmic Aspects of Comparability Graphs and Interval Graphs. In I. Rival, editor, *Graphs and Order : The Role of Graphs in the Theory of Ordered Sets and Its Applications*, NATO ASI Series, pages 41–101. Springer Netherlands, Dordrecht, 1985.

[81] Z. Pawlak. Rough sets. *International Journal of Computer & Information Sciences*, 11(5) :341–356, October 1982.

[82] R. Potharst and J.C. Bioch. Decision trees for ordinal classification. *Intelligent Data Analysis*, 4(2) :97–111, January 2000.

[83] H. Prade, A. Rico, M. Serrurier, and E. Raufaste. Elicitating Sugeno Integrals : Methodology and a Case Study. In C. Sossai and G. Chemello, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertain*, volume 5590 of *Lecture Notes in Computer Science*, pages 712–723, Verona, Italy, July 2009. Springer-Verlag, Berlin, Germany.

[84] J.R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[85] M. Rademaker, B. De Baets, and H. De Meyer. Loss optimal monotone relabeling of noisy multi-criteria data sets. *Information Sciences*, 179(24) :4089–4096, December 2009.

[86] A. Rico, M. Grabisch, C. Labreuche, and A. Chateauneuf. Preference modeling on totally ordered sets by the Sugeno integral. *Discrete Applied Mathematics*, 147(1) :113–124, April 2005.

[87] M. Roubens. Ordinal Multiattribute Sorting and Ordering in the Presence of Interacting Points of View. In D. Bouyssou, E. Jacquet-Lagrèze, P. Perny, R. Słowiński, D. Vanderpooten, and P. Vincke, editors, *Aiding Decisions with Multiple Criteria : Essays in Honor of Bernard Roy*, International Series in Operations Research & Management Science, pages 229–246. Springer US, Boston, MA, 2002.

[88] L.S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28) :307–317, 1953.

[89] M. Siggers. On the representation of finite distributive lattices. *arXiv preprint arXiv :1412.0011*, 2014.

[90] J. Sill. Monotonic networks. In *Advances in neural information processing systems*, pages 661–667, 1998.

[91] M. Sugeno. *Theory of fuzzy integrals and its applications*. Tokyo Institute of Technology, 1974.

[92] R.E. Tarjan and A.E. Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing*, 6(3) :537–546, 1977.

[93] A.F. Tehrani. *Learning monotone nonlinear classifier using the Choquet integral*. PhD thesis, Philipps-Universität Marburg, 2014.

[94] A.F. Tehrani, W. Cheng, K. Dembczyński, and E. Hüllermeier. Learning monotone nonlinear models using the Choquet integral. *Machine Learning*, 89(1) :183–211, 2012.

[95] A.F. Tehrani, W. Cheng, and E. Hullermeier. Preference Learning Using the Choquet Integral : The Case of Multipartite Ranking. *IEEE Transactions on Fuzzy Systems*, 20(6) :1102–1113, 2012.

[96] A.F. Tehrani and E. Hüllermeier. Ordinal Choquistic Regression. page 8, 2013.

[97] R. Van De Kamp, A.J. Feelders, and N. Barile. Isotonic Classification Trees. In *Advances in Intelligent Data Analysis VIII*, Lecture Notes in Computer Science, pages 405–416. Springer, Berlin, Heidelberg, August 2009.

[98] V.V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin Heidelberg, 2003.

[99] A. Verkeyn, D. Botteldooren, B. De Baets, and G. De Tré. Sugeno Integrals for the Modelling of Noise Annoyance Aggregation. In T. Bilgiç, B. De Baets, and O. Kaynak, editors, *Fuzzy Sets and Systems — IFSA 2003*, Lecture Notes in Computer Science, pages 277–284. Springer Berlin Heidelberg, 2003.

[100] X-Z. Wang, Y-L. He, L-C. Dong, and H-Y. Zhao. Particle swarm optimization for determining fuzzy measures from data. *Information Sciences*, 181(19) :4230–4252, October 2011.

[101] Z. Wang, K-S. Leung, and J. Wang. Determining nonnegative monotone set functions based on Sugeno's integral : an application of genetic algorithms. *Fuzzy Sets and Systems*, 112(1) :155–164, May 2000.

[102] I. Wegener. *The complexity of Boolean functions*. John Wiley & Sons, Inc., August 1987.

[103] H. Zhu, E.C.C. Tsang, X.Z. Wang, and R. Aamir Raza Ashfaq. Monotonic classification extreme learning machine. *Neurocomputing*, 225 :205–213, February 2017.

Résumé

Une Fonction Latticielle Polynômiale (FLP) sur un treillis L est une fonction $p : L^n \to L$, qui peut être exprimée à partir de variables, de constantes et des opérateurs de treillis $\wedge$ et sup. Dans les cas où L est distributif et borné, les FLP incluent les intégrales de Sugeno. Celles-ci sont des fonctions d'agrégation qui permettent de fusionner des valeurs sur des échelles ordinales non numériques, et qui sont utilisées notamment dans l'approche qualitative de l'Aide à la Décision Multi Critères (MCDA) en tant qu'alternatives ordinales aux intégrales de Choquet.

Dans une première partie, nous traitons la tâche d'interpolation par des FLP, c'est à dire : pour un treillis L, un sous-ensemble fini D de L^n et une fonction $f : D \to L$, retourner une FLP $p : L^n \to L$ telle que $p(\mathbf{x}) = f(\mathbf{x})$ pour tout $\mathbf{x} \in D$ (si une telle FLP existe). Nous traitons successivement le cas où L est un treillis fini et le cas où L est une treillis distributif borné. Dans les deux cas, nous donnons des algorithmes qui résolvent ce problème en temps polynomial.

Dans une seconde partie, nous abordons les généralisations des intégrales de Sugeno appelées Fonctions d'Utilité de Sugeno (FUS), qui permettent la fusion de valeurs appartenant à des échelles ordinales différentes, ainsi que leur application à la tâche de classification monotone. Nous introduisons un modèle composé de plusieurs FUS, ainsi qu'un algorithme d'apprentissage d'un tel modèle. Nous comparons ce modèle aux ensembles de règles de décision appris par VC-DomLEM, et étudions le nombre de FUS nécessaires afin de modéliser des données empiriques.

Mots-clés: polynôme latticiel, intégrale de Sugeno, agrégation, interpolation, classification monotone, règles de décision

Abstract

A Lattice Polynomial Function (LPF) over a lattice L is a map $p : L^n \to L$ that can be defined by an expression involving variables, constants and the lattice operators $\wedge$ and $\vee$. If L is a distributive lattice, these maps include the so-called Sugeno integrals that are aggregation functions capable of merging ordinal values, not necessarily numerical. They are widely used in the qualitative approach to Multiple Criteria Decision Aiding (MCDA), and they can be thought of as the ordinal counterparts of Choquet integrals.

In the first part of this thesis, we tackle the task of interpolating a partial function by an LPF, stated as follows : for a lattice L, a finite subset D of L^n, and a function $f : D \to L$, return an LPF $p : L^n \to L$ such that $p(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in D$ (if such an LPF exists). We treat the cases where L is a finite lattice, and then the cases where L is a bounded distributive lattice. In both cases, we provide algorithms that solve this problem in polynomial time.

In the second part, we consider generalizations of Sugeno integrals in the multi-attribute setting, in particular, the Sugeno Utility Functions (SUFs), that are able to merge values coming from different ordinal scales. We consider the their use in monotonic classification tasks. We present a model based on a set of SUFs and an algorithm for learning such model from data. We compare this model to the sets of monotonic decision rules learned by VC-DomLEM, and study the number of SUFs that are required in order to model empirical data.

Keywords: lattice polyomial, Sugeno integral, aggregation, interpolation, monotonic classification, decision rules