Apprentissage symbolique à partir de données issues de simulation pour l'aide à la décision. Gestion d'un bassin versant pour une meilleure qualité de l'eau.

Ronan Trépos

N° d'ordre: 3696

THÈSE

Présentée devant

devant l'Université de Rennes 1

pour obtenir

le grade de : Docteur de l'Université de Rennes 1
Mention Informatique

par

Ronan TRÉPOS

Équipe d'accueil : DREAM - IRISA
École Doctorale : Matisse
Composante universitaire : IFSIC

Titre de la thèse :

Apprentissage symbolique à partir de données issues de simulation pour l'aide à la décision.
Gestion d'un bassin versant pour une meilleure qualité de l'eau.

soutenue le 22 janvier 2008 devant la commission d'examen

Rapporteurs :	Christel Vrain	PR. Université d'Orléans
	Robert Faivre	DR. INRA de Toulouse
Examinateurs :	Jocelyne Erhel	DR. INRIA Rennes
	Yves Le Bissonnais	DR. INRA de Montpellier
Directrice de thèse :	Marie-Odile Cordier	PR. Université de Rennes 1
Co-encadrantes :	Chantal Gascuel-Odoux	DR. INRA de Rennes
	Véronique Masson	MdC. Université de Rennes 1

Avant-Propos

Cette thèse a été encadrée par une équipe composée de Marie-Odile Cordier (IRISA, Rennes), Frederik Garcia (UMR MIA, INRA, Toulouse), Chantal Gascuel-Odoux (UMR SAS, INRA, Rennes) et Véronique Masson (IRISA, Rennes). Elle a été accompagnée par un comité de pilotage de thèse composé de J. Baudry, P. Durand, F. Le Berre et C. Vrain, qui s'est réuni une fois par an.

Cette thèse a été effectuée dans l'équipe DREAM de l'IRISA. Elle a été co-financée par les départements "Environnement et Agronomie" et "Mathématique Appliquée et Informatique" de l'INRA.

Ce travail fait partie du projet SACADEAU "Système d'Acquisition de Connaissance pour d'Aide à la Décision pour la qualité de l'EAU", qui a répondu à l'Action transversale INRA-CIRAD "Aide à la Décision : comment articuler connaissances et action en agriculture, agroalimentaire et dans l'espace rural". Ce projet a associé des équipes de recherche et des partenaires des chambres d'agricultures de Bretagne, en particulier L. Lebouille, animatrice du bassin versant du Frémeur, M. Falchier et D. Heddadj.

Remerciements

Je tiens à remercier Christel Vrain et Robert Faivre pour avoir accepté de rapporter cette thèse. Leurs remarques ont été précieuses. Je remercie aussi Jocelyne Erhel et Yves Le Bissonnais d'avoir également participé au jury de cette thèse.

Merci beaucoup à Marie-Odile-Cordier, Chantal Gascuel-Odoux et Véronique Masson pour leur encadrement. Elles ont su orienter ces travaux tout en me laissant une grande liberté dans mes choix, ce qui a été très appréciable. Elles ont été disponibles et ont su me soutenir tout au long de ces trois années.

Je remercie également Fred Garcia pour son investissement et ses retours sur cette thèse. Je remercie aussi les membres du projet SACADEAU, et ceux du comité de thèse, qui m'ont apporté des éléments sur la problématique environnementale de ce travail.

L'accueil au sein de l'équipe DREAM a été remarquable. Entre autres, la bonne humeur de Sophie (qui va de pair avec celle de son amie Véro), les jeux de Philippe, les conseils techniques de René ont été à l'origine de bons moments passés en leur compagnie, par exemple dans le cadre des séminaires Tahiti, toujours très bien organisés d'ailleurs :-) Un grand merci également à Elisa, François et Alban, jeunes docteurs maintenant, qui m'ont aidé à comprendre ce qui constitue un travail de thèse.

Je tiens à remercier Lucie, Goulven, Xavier et Alexandre pour les bons moments passés en leur compagnie à l'Irisa. Une dédicace particulière à Alex, colloc de bureau pendant ces trois années. Pour les bons moments passés à Rennes pendant ces trois années, merci à tous ceux que j'ai côtoyés, par exemple à l'UBU avec entre autres Sasha, Arnold et David ; au poker, et un peu partout avec des amis de longue date, les Mathieu, Bertrand, Damien, Yann...

Un grand merci enfin adressé à mes parents, mes frères et leur compagne pour m'avoir soutenu, réellement, tout au long de ces trois années. A tout moment, ils m'ont encouragé.

Table des matières

Liste des Définitions

Liste des Algorithmes

Liste des Tableaux

Liste des Figures

Liste des Exemples

Introduction

Contexte

L'évaluation de l'impact des activités humaines sur les ressources en eau, le sol ou la biodiversité, est un enjeu important pour la préservation de l'environnement. Les interactions entre ces activités humaines et les processus biophysiques sont nombreuses, ce qui rend cette évaluation, et en réponse les recommandations pour une meilleure gestion des activités humaines, difficiles. Les systèmes environnementaux sont en général caractérisés par des données peu nombreuses et une certaine méconnaissance des processus, alors qu'ils font l'objet d'un questionnement sur leur évolution. De nombreux modèles ont été développés. Un premier objectif de ces modèles peut être de confirmer ou infirmer une hypothèse de recherche sur l'importance des processus mis en jeu. Un autre objectif est le développement de modèles à des fins opérationnelles, par exemple pour l'aide à la gestion d'un territoire. Ces modèles sont en général complexes car ils couplent différents modèles. Leurs entrées et sorties peuvent être nombreuses et exprimées sous plusieurs formes : il s'agit par exemple de variables qualitatives ou quantitatives, de relations spatiales et temporelles.

Interpréter les résultats d'un modèle de simulation est difficile en raison du grand nombre d'entrées-sorties et de l'interaction forte entre les modèles couplés. Pourtant, l'interprétation des résultats est indispensable si le modèle est destiné à l'aide à la prise de décision. Une démarche parfois proposée est la simulation de scénarios, qui expriment des questions du type "que se passerait-il si" ou plus précisément comment évoluent les sorties si les entrées varient d'une certaine façon. La démarche développée dans ce travail de thèse propose d'utiliser des méthodes d'apprentissage symbolique. En particulier, nous nous intéressons à l'apprentissage automatique de relations entre les entrées et les sorties d'un modèle de simulation. L'apprentissage symbolique permet l'induction de règles décrivant l'impact, sur les sorties de simulation, de facteurs pertinents pour la prise de décision. Les objectifs sont, en s'appuyant sur l'analyse des règles induites, d'une part de dégager les variables explicatives des sorties de simulation, d'autre part de suggérer des actions permettant d'améliorer une situation.

Contributions

L'induction de règles de classification consiste à découvrir, à partir d'observations étiquetées par une variable qualitative appelée classe, des règles du type *si* <*condition*> *alors* <*classe*> où les conditions portent sur des observations. Dans le cadre de notre travail, les observations décrivent les entrées du simulateur et la classe est associée à ses sorties. Une observation et son étiquette constituent un exemple d'apprentissage, une règle de classification est une généralisation des exemples.

Nous nous sommes intéressés à des exemples d'apprentissage qui sont des arbres dont les nœuds sont décrits par un ensemble d'attributs. Deux approches sont proposées pour l'induction des règles. La première consiste à générer des motifs d'arbres, ou sous-arbres, contenant des contraintes sur les valeurs des attributs aux nœuds. Cette approche nécessite une description des exemples et des règles dans un langage relationnel. Nous avons défini un opérateur de spécialisation de tels motifs afin de mettre en œuvre, au sein du système de Programmation Logique Inductive Aleph, une recherche descendante dans l'espace des hypothèses. La deuxième approche repose sur la définition experte d'attributs synthétisant l'information contenue dans les structures d'arbre. Cela nous permet de décrire les exemples dans une logique attribut-valeur. Le système d'apprentissage de règles CN2 est ensuite utilisé pour l'induction des règles de classification dans le cadre de cette logique. Nous avons comparé les deux approches sur les données de notre application.

Lorsque le nombre de règles induites est important, il est intéressant de proposer des méthodes d'analyse de ces règles. Étant donnés deux classes $\oplus$ et $\ominus$, un ensemble de règles de classification dans une logique attribut-valeur et une situation S jugée non satisfaisante par l'utilisateur (par exemple étiquetée par la classe $\ominus$), nous avons défini une tâche de recommandation d'actions. L'objectif est de proposer des actions, définies comme des modifications de la valeur des attributs décrivant S, qui appliquées à S, permettent d'améliorer sa classe. Nous nous sommes intéressés aux notions de faisabilité et de qualité d'une action et avons proposé un espace de recherche des actions. Deux algorithmes ont été développés, puis comparés, pour générer les actions. Le premier repose sur une recherche par faisceau en s'appuyant sur une spécialisation des actions. Le second génère des actions complexes, définies à partir des sous-ensembles de règles de classe $\oplus$, et permet dans certaines conditions de découvrir une action optimale, pour le critère de qualité, dans l'espace de recherche.

Ces contributions ont été motivées et expérimentées par l'application de gestion d'un bassin versant en vue d'améliorer la qualité de l'eau qui est l'objet du projet SACADEAU.

Application

La contamination des eaux de surface par les pesticides est un des problèmes majeurs de la ressource en eau. En Bretagne en 2005, une étude réalisée par la DRASS (Direction Régionale des Affaires Sanitaires et Sociales de Bretagne) montre que 80% des échantillons d'eau de surface effectués dépassent les seuils $0,1\ \mu g.l^{-1}$ pour une seule

substance active[1]. Il est très fréquent en Bretagne que les seuils de potabilité de l'eau, en terme de concentrations, soient dépassés. Des fuites très faibles, inférieures à 1% des quantités appliquées, suffisent le plus souvent à entraîner un dépassement des seuils de potabilité. Les processus de transfert de pesticides sont suffisamment connus pour être simulés au niveau d'une parcelle agricole. Par contre, lors du passage à l'échelle d'un territoire plus important, tel qu'un bassin versant, les résultats de simulation sont difficiles à interpréter. En effet beaucoup de facteurs entrent en jeu comme la localisation spatiale et temporelle des applications de pesticides dans le cas du désherbage de cultures de maïs.

Le projet SACADEAU vise à développer un outil pour l'aide à la prise de décision des personnes en charge d'un territoire dans la préservation de la qualité de l'eau. L'application cible concerne la gestion des activités agricoles et des aménagements sur un bassin versant (territoire drainé par une rivière) et le problème du transfert de pesticides, depuis son application sur une culture de maïs jusqu'à la rivière. La première étape du projet a été le développement d'un modèle de simulation du transfert au sein d'un bassin versant. Ce modèle est le couplage de deux modèles, l'un simulant les pratiques agricoles pour la culture du maïs, l'autre simulant le transfert des pesticides. La deuxième étape du projet SACADEAU, qui constitue ce travail de thèse, est l'analyse des résultats de simulation par des méthodes d'apprentissage symbolique pour l'aide à la décision.

En s'appuyant sur la représentation interne des écoulements par ruissellement du modèle, nous proposons une représentation spatiale du bassin versant par un ensemble d'*arbres d'exutoires*. Un arbre d'exutoires est une représentation adaptée pour, d'une part simuler le transfert, et d'autre part représenter les pratiques agricoles. Il s'agit d'un "sous-bassin versant" représenté par une structure d'arbre où chaque nœud est une partie de parcelle, qui est l'unité fonctionnelle pour les pratiques agricoles, et où chaque arc est une relation d'écoulement entre les nœuds. Un arc représente une relation spatiale entre deux parcelles, l'une à l'amont, l'autre à l'aval. Nous nous intéressons alors au taux de transfert d'un arbre d'exutoires, qui est le ratio entre les quantités de pesticides appliquées et celles transférées. Un arbre d'exutoires représente un exemple d'apprentissage dont l'étiquette est calculée à partir du taux de transfert. Une base d'exemple est constituée à partir des résultats de simulation du modèle. L'induction des règles permet de faire émerger les relations d'écoulements pertinentes pour le transfert ainsi que l'impact des facteurs liés à la prise de décision. Afin d'aider à l'analyse des règles induites, une visualisation des exemples et règles est proposée. Enfin, une méthode de recommandation d'action s'appuie sur les règles induites et suggère des actions à effectuer pour améliorer une situation de transfert important.

[1] Un décret du 20 décembre 2001 fixe la limite des concentrations pour la potabilité de l'eau à $0,1$ $\mu g.l^{-1}$ pour un seul pesticide et à $0,5$ $\mu g.l^{-1}$ pour l'ensemble des pesticides mesurés. Les résultats donnés ici pour la Bretagne proviennent d'une étude réalisée par la DRASS `http://www.draf.bretagne.agriculture.gouv.fr/corpep/IMG/pdf/DRASS_pesticides_2005_CORPEP.pdf`

Plan

Ce mémoire se divise en quatre chapitres. Le premier chapitre introduit l'apprentissage de règles dans le cadre des deux logiques utilisées pour la caractérisation du taux de transfert des arbres d'exutoires. Les deux logiques sont la logique attribut-valeur et la logique des prédicats ; cette dernière est introduite dans l'annexe A.

Le deuxième chapitre présente le modèle SACADEAU et les différents types de transfert qui sont modélisés. Les résultats de simulation de scénarios y sont également présentés. Ces scénarios sont définis de manière à évaluer l'impact de différents facteurs.

Le troisième chapitre présente les différents apprentissages de règles mis en œuvre entre les entrées et les sorties du modèle SACADEAU et se termine par une comparaison de ces apprentissages.

Le dernier chapitre présente l'analyse de l'ensemble des règles induites, c'est à dire la recommandation d'action pour une situation jugée insatisfaisante, à partir de la théorie induite et la visualisation proposée des règles et des exemples.

Chapitre 1

Apprentissage de règles

L'apprentissage artificiel consiste à construire automatiquement des hypothèses permettant d'expliquer des observations. Étant donné un ensemble d'observations, appelé par la suite ensemble d'exemples ou ensemble d'apprentissage, il s'agit de trouver une théorie $HSet$, c'est à dire une ou plusieurs hypothèses, qui permette d'expliquer les exemples.

Dans un processus d'apprentissage (Mitchell, 1982), le langage des exemples $\mathcal{L}_E$ et celui des hypothèses $\mathcal{L}_H$ doivent être définis. Ceux-ci permettent de donner en intention l'espace des exemples $\mathcal{E}_E$ et l'espace des hypothèses $\mathcal{E}_H$. De plus une relation de couverture, $covers : \mathcal{E}_H \rightarrow \mathcal{P}(\mathcal{E}_E)$ (où $\mathcal{P}(\mathcal{E}_E)$ est l'ensemble des parties de $\mathcal{E}_E$), doit être définie entre une hypothèse et une partie des exemples. Pour une hypothèse $H \in \mathcal{E}_H$, $covers(H)$ est l'ensemble des exemples que H explique ; on dit aussi que H couvre les exemples $covers(H)$.

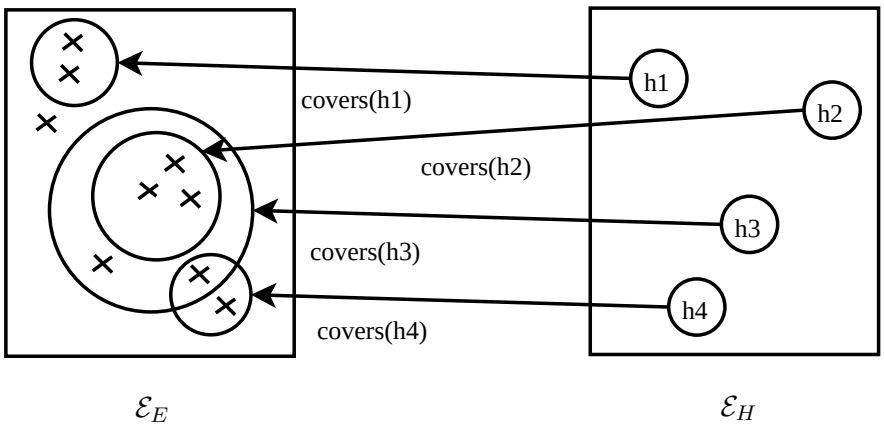


FIG. 1.1 – Un problème d'apprentissage selon Mitchell (Mitchell, 1982) : construire des hypothèses de l'espace des hypothèses $\mathcal{E}_H$ (à droite) en s'aidant de la relation de couverture qui lie l'espace des exemples $\mathcal{E}_E$ (à gauche) et $\mathcal{E}_H$. Dans l'espace des hypothèses, idéalement, $h3$ doit être une généralisation de $h2$ car $covers(h2) \subseteq covers(h3)$.

La figure 1.1 est une représentation du problème d'apprentissage artificiel. Effectuer un apprentissage artificiel consiste à obtenir les observations $E \subseteq \mathcal{E}_E$ (constituer une base d'apprentissage), les exprimer dans le langage des exemples $\mathcal{L}_E$ qu'il faut définir, définir aussi le langage des hypothèses $\mathcal{L}_H$ et accomplir une recherche dans l'espace des hypothèses. Les choix effectués au cours de ces différentes étapes peuvent être dépendants les uns des autres.

Afin de guider la recherche dans l'espace des hypothèses, on munit celui-ci d'une relation de *généralité* ($\succeq$). Idéalement, une hypothèse H_1 est une *spécialisation* d'une hypothèse H_2, ou H_2 est une *généralisation* de H_1, si $covers(H_1) \subseteq covers(H_2)$ (figure 1.1).

Nous nous restreignons dans cette thèse à l'apprentissage de *règles de classification*. Cela signifie d'une part que chaque exemple est étiqueté par une valeur particulière appelée la classe de l'exemple ; nous parlons alors d'apprentissage *supervisé*. En se restreignant à deux classes, les étiquettes $\oplus$ et $\ominus$, l'ensemble d'apprentissage est composé des ensembles $E^{\oplus}$ et $E^{\ominus}$ que l'on appelle exemples positifs et négatifs. Si l'on veut expliquer les exemples de classe $\oplus$, il s'agit alors de construire des hypothèses qui couvrent des exemples de $E^{\oplus}$ et aucun de $E^{\ominus}$. Pour comparaison, le problème d'apprentissage de *règles d'associations*, exprimé dans l'exemple 1.2 est un problème d'apprentissage non supervisé.

Exemple 1.2 (la recherche de règles d'association).
L'exemple typique du problème de recherche de règles d'association est le problème du panier de la ménagère : dans une grande surface commerciale par exemple, il s'agit de retrouver l'achat de certains produits qui influent l'achat d'autres produits. On s'intéresse aux règles du type : *si un client achète les produits* p_1 *et* p_2 *alors il achète aussi les produits* p_3 *et* p_4. Formellement cette règle s'écrit : $p_1, p_2 \rightarrow p_3, p_4$. Un exemple d'apprentissage correspond aux achats d'un client. C'est un problème d'apprentissage non supervisé, les exemples ne sont pas étiquetés par une classe. Le système APRIORI (Agrawal *et al.*, 1993) a rendu populaire la recherche de règles d'associations.

D'autre part, une règle de classification peut être notée sous la forme *si <conditions> alors <classe=c>* et qui signifie "si un exemple respecte les conditions *conditions* alors sa classe est c". Une théorie, c'est à dire un ensemble de règles de classifications, respecte dans le cas idéal les conditions de *correction* et *complétude* (Michalski *et al.*, 1986b).

Définition 1.3 (Complétude et correction d'une théorie).
Soient une théorie HSet et un ensemble d'exemples E, on note classe(o) la classe de o, où o est soit une règle de HSet, soit un exemple de E.

- *HSet est dite complète si :* $\forall e \in E, \exists r \in HSet, e \in covers(r)$;
- *HSet est dite correcte si :* $\forall e \in E, \forall r \in HSet$, *si* $e \in covers(r)$ *alors* $classe(e) = classe(r)$. *On dit également qu'une règle de classe c est correcte si elle ne couvre que des exemples de classe c.*

Une théorie est complète si tous les exemples sont expliqués par au moins une de ses règles, elle est correcte si il n'existe pas de règle d'une certaine classe couvrant un exemple d'une autre classe.

La spécification du langage $\mathcal{L}_H$ définit le type de conditions qui sont dans la partie gauche de la règle. Dans ce chapitre, nous nous intéressons à deux langages pour définir $\mathcal{L}_E$ et $\mathcal{L}_H$: la logique attribut-valeur et la logique des prédicats. Pour chacun d'eux, nous donnons un aperçu des méthodes d'induction de règles.

1.1 Apprentissage de règles attribut-valeur

Dans le contexte de l'apprentissage de règle attribut-valeur, une description des exemples d'apprentissage est réalisée par un ensemble d'attributs $\mathcal{X}$. Pour chaque attribut $A \in \mathcal{X}$ un *domaine* $Dom(A)$ est défini, il constitue l'ensemble des valeurs qui peuvent être prises par l'attribut. Un attribut peut être soit quantitatif, soit qualitatif (exemple 1.4).

Pour une règle attribut-valeur, Michalski (Michalski, 1973) a introduit le terme *sélecteur* pour décrire les conditions d'une règle. Un *sélecteur* est sous la forme $A\#const$, où $A \in \mathcal{X}$ est un attribut et $const \in Dom(A)$ est une constante du domaine de A. Le terme $\#$ désigne un comparateur dans l'ensemble $\{=, \neq, >, \geq, <, \leq\}$. Les comparateurs $>, \geq, <$ et $\leq$ ne peuvent être utilisés que lorsqu'une relation d'ordre est définie sur le domaine de A, ce qui est le cas, par exemple, des attributs quantitatifs.

Par la suite nous utiliserons la notation de Clark (Clark & Niblett, 1989) ; en particulier, un *complexe* est une conjonction de sélecteurs. Le complexe formé de la conjonction des sélecteurs $s_1, s_2, \ldots, s_n$ s'écrit $s_1 \wedge s_2 \wedge \ldots \wedge s_n$. Dans l'apprentissage de règles de classification attribut-valeur, nous restreignons le langage des hypothèses $\mathcal{L}_H$ aux hypothèses qui s'écrivent sous la forme *si complex alors classe=c* où *complex* désigne un complexe constitué des sélecteurs sur des attributs de $\mathcal{X}$. La classe est souvent vue comme un attribut qualitatif particulier pour décrire un exemple mais que l'on retrouve uniquement et obligatoirement dans la partie droite de la règle : on cherche finalement à expliquer la classe des exemples en fonction des attributs de $\mathcal{X} - \{classe\}$.

Les exemples d'apprentissage sont décrits par des complexes constitués uniquement de sélecteurs du type $A = const$, c'est-à-dire construits avec le seul comparateur $=$. Un exemple est couvert par une règle H si pour un sélecteur sel, $A\#const_H$ décrivant H, la constante $const$ du sélecteur $A = const$ décrivant l'exemple pour le même attribut respecte la contrainte décrite par le sélecteur sel (exemple 1.4). La relation de couverture *covers* relie à H l'ensemble des exemples de E couverts par H. Pour une règle H sous la forme *si complex alors classe=c*, on notera également $covers(complex)$ qui désignera l'ensemble $covers(H)$. En général, la relation de *généralisation* est définie par l'inclusion des sélecteurs d'une règle dans une autre. Autrement dit, une règle $r1$ est une spécialisation d'une règle $r2$, ou $r2$ est plus générale que $r1$, si tous les sélecteurs du complexe gauche de $r2$ sont présents dans le complexe gauche de $r1$. On a bien la propriété souhaitée qui est : si $r2$ est plus générale que $r1$ alors $covers(r1) \subseteq covers(r2)$.

Exemple 1.4 (relation de généralité entre règles attribut-valeur).
On considère l'attribut *poids* dont le domaine non fini est celui des réels positifs et l'attribut *couleur* de domaine fini $\{rouge, vert\}$. L'attribut *poids* est un attribut quantitatif et l'attribut *couleur* est un attribut qualitatif. La *classe* prend ses valeurs dans

$\{c_1, c_2\}$. Les sélecteurs $poids \geq 50$ et $classe = c_1$ permettent de former la règle :

$$poids \geq 50 \rightarrow classe = c_1$$

Cette règle est plus générale que la règle :

$$poids \geq 50 \wedge couleur = rouge \rightarrow classe = c_2$$

En effet, tous les sélecteurs du complexe gauche de la première règle sont présents dans le complexe gauche de la deuxième. L'exemple d'apprentissage décrit par le complexe $poids = 70 \wedge couleur = vert$ est couvert par la première règle mais pas par la seconde car les sélecteurs $couleur = vert$ et $couleur = rouge$ sont en conflit.

De nombreux systèmes d'apprentissage cherchent à induire des théories en logique attribut-valeur. Un des premiers systèmes développés a été ID3 (Quinlan, 1986) ; il construit en fait des arbres de décision (figure 1.5) qui, a posteriori, peuvent facilement être traduits en ensembles de règles de classification (sous-section 1.1.1). Le système CN2 (Clark & Niblett, 1989) recherche directement les règles de classification (sous-section 1.1.2). Ces deux systèmes sont représentatifs de stratégies différentes pour l'induction de théories (sous-section 1.1.3).

1.1.1 Les arbres de décision

Un arbre de décision est un classifieur dont la structure est comme son nom l'indique un arbre. La figure 1.5 en montre un exemple. Chaque nœud N de l'arbre, hormis les feuilles, représente un attribut A. Les feuilles représentent quant à elles des classes d'apprentissage. Chaque arc de l'arbre dont la source est N représente un test sur l'attribut A, comportant un comparateur $\#$ et une constante *const* du domaine de A. Un arc et sa source forment le sélecteur $A\#const$.

Pour un nouvel exemple dont on ne connaît pas l'étiquette, on parcourt l'arbre du nœud racine vers une feuille et on affecte ainsi à l'exemple la classe de la feuille atteinte.

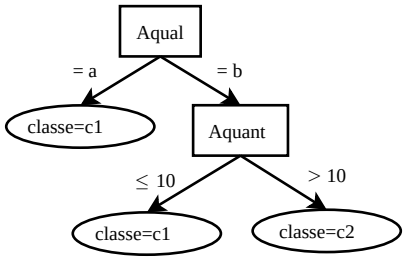


FIG. 1.5 – Dans cet arbre de décision, la racine représente l'attribut qualitatif *Aqual* dont le domaine est $\{a, b\}$. L'attribut *Aquant* dont le domaine est $\Re$ sert à classer les exemples pour lesquels l'attribut *Aqual* prend la valeur b. Les trois règles générées à partir de cet arbre sont : *si* $Aqual = a$ *alors classe=c1*, *si* $Aqual = b \wedge Aquant \leq 10$ *alors classe=c1* et enfin *si* $Aqual = b \wedge Aquant > 10$ *alors classe=c2*

Algorithme 1.6 : Algorithme de construction d'un arbre de décision

Entrées : - un ensemble d'exemples E
Sorties : - un arbre de décision
si *tous les exemples de* E *sont de classe* c **alors**
  retourner une feuille $classe = c$
sinon
  choisir un attribut A
  construire une racine r représentant A
  si A *est quantitatif* **alors**
    choisir une valeur $v \in Dom(A)$
    soit E' les exemples qui vérifient $A \leq v$
    construire un arbre de décision pour E' et le rattacher à r
    soit E'' les exemples qui vérifient $A > v$
    construire un arbre de décision pour E'' et le rattacher à r
  sinon
    pour chaque *valeur* v *de* $Dom(A)$ **faire**
      soit E' les exemples qui vérifient $A = v$
      construire un arbre de décision pour E' et le rattacher à r
retourner l'arbre de décision de racine r

Un arbre de décision se construit de manière récursive, depuis sa racine jusqu'à ses feuilles. Pour un nœud N, il faut spécifier s'il est une feuille ou non. S'il est une feuille, il faut déterminer de quelle classe d'apprentissage il s'agit, sinon il faut déterminer quel est l'attribut A qui lui est rattaché et les différents sélecteurs qui forment les arcs dont N est la source. Les comparateurs utilisés pour la génération des sélecteurs sont en général $\{\leq, >\}$ si l'attribut A est quantitatif et $\{=\}$ s'il est qualitatif (voir algorithme 1.6).

La construction d'un arbre de décision nécessite l'utilisation d'heuristiques pour choisir l'attribut A attaché au nœud N et, si A est quantitatif, pour déterminer les constantes de $Dom(A)$ attachés aux arcs dont N est la source. Pour plus de détails, les heuristiques couramment utilisées telles que *l'entropie croisée* sont données dans (Cornuéjols & Miclet, 2002). Les méthodes d'élagage de l'arbre utiles, pour une meilleure généralisation des exemples, y sont également abordées. L'algorithme précurseur des arbres de décision est ID3 (Quinlan, 1986). Les systèmes C4.5 (Quinlan, 1993) et CART (Breiman *et al.*, 1984) sont très utilisés aujourd'hui.

La génération d'une théorie $HSet$ à partir d'un arbre de décision se fait de manière assez naturelle (Quinlan, 1987). Une règle de classification attribut-valeur est décrite par un parcours depuis la racine jusqu'à une feuille. Le complexe est formé par l'ensemble des sélecteurs rencontré lors de ce parcours et la classe de la règle est celle donnée par la feuille (voir l'exemple en figure 1.5).

1.1.2 Le système CN2

Le système CN2 (Clark & Niblett, 1989; Clark & Boswell, 1991) recherche un ensemble de règles de classification attribut-valeur $HSet$ pour généraliser des exemples E. Nous nous appuyons ici sur l'article le plus récent (Clark & Boswell, 1991) pour présenter ce système. Dans cette variante, une théorie est construite pour chacune des classes d'apprentissage.

Lors du parcours de l'espace des hypothèses $\mathcal{E}_H$, CN2 utilise deux heuristiques pour évaluer les hypothèses et les comparer entre elles. Notons, pour un problème d'apprentissage à k classes, *comp* un complexe, n le nombre total d'exemples, n^c le nombre d'exemples de classe c, $n(comp)$ le nombre d'exemples couverts par *comp* et $n^c(comp)$ le nombre d'exemples de classe c couverts par *comp*.

En s'inspirant des travaux de Niblett (Niblett, 1987), Clark et Boswell s'appuient sur l'*estimateur de Laplace* pour évaluer une règle. Il est défini par :

$$laplace(comp, c) = \frac{n^c(comp) + 1}{n(comp) + k} \tag{1.7}$$

Un estimateur de Laplace élevé indique que la règle construite à partir du complexe *comp* pour la classe c couvre essentiellement des exemples de classe c, cette règle est donc un bon candidat. Notons que si $n^c(comp) \neq n(comp)$, la condition de correction (définition 1.3) est relâchée puisque *comp* couvre des exemples d'autres classes.

La deuxième heuristique utilisée est le *rapport de vraisemblance* (de l'anglais *likelihood ratio*) (Kalbfleish, 1979). Il est défini par :

$$rapport\_vraisemblance(comp) = 2 * \sum_{c=1,..,k} f_c * log(\frac{f_c}{p_a(c)})$$

Où f_c et $p_a(c)$ sont définis de la manière suivante[1] : $f_c = n^c(comp)/n(comp)$ et $p_a(c) = n^c/n$. Un rapport de vraisemblance élevé indique que *comp* est significatif ; en fait, il compare les distributions (a priori) dans les différentes classes de tous les exemples et ces distributions (a posteriori) pour les exemples couverts par *comp* uniquement.

L'induction d'une règle pour une classe c au sein du système CN2, s'appuie sur une *recherche par faisceau* (voir algorithme 1.8). Il maintient un ensemble $faisceau$ d'hypothèses (ici des complexes). Une étape est la construction de nouveaux complexes en ajoutant des sélecteurs aux complexes du faisceau. Ces nouveaux complexes sont donc des spécialisations des anciens. Le faisceau est alors mis à jour avec les meilleurs nouveaux complexes, au sens de l'estimateur de Laplace, et la taille du faisceau est bornée par un paramètre TAILLE_MAX_FAISCEAU. Au cours de la recherche, le meilleur complexe, toujours au sens de l'estimateur de Laplace, dont le rapport de vraisemblance est supérieur à un paramètre MIN_VRAI est conservé.

Enfin pour construire un ensemble de règles $HSet$ pour une classe c, le système adopte une stratégie de type *"séparer pour régner"* (voir l'algorithme 1.9). Il construit une première règle H pour la classe c qu'il ajoute à $HSet$ en utilisant l'ensemble des

[1] Dans le cas où $n(comp)$ est égal à 0, f_c est défini comme étant égal à 0.

Algorithme 1.8 : Algorithme de CN2 pour la construction d'une règle

Entrées : - un ensemble d'exemples E
- une classe c

Sorties : - une règle H

$faisceau_1 := \{complexe\_vide\}$
$meilleur\_complexe := nul$
tant que $faisceau_1 \neq \emptyset$ **faire**
  $faisceau_2 = \emptyset$
  pour chaque *complexe* $comp \in faisceau_1$ **faire**
    pour chaque *sélecteur sel non présent dans comp* **faire**
      $complexe := comp \wedge sel$
      $faisceau_2 := faisceau_2 \cup \{complexe\}$
      si $laplace(complexe, c) > laplace(meilleur\_complexe, c)$ *et*
      $rapport\_vraisemblance(complexe) >$ *MIN_VRAI* **alors**
        $meilleur\_complexe := complexe$
      si $taille(faisceau_2) >$ *TAILLE_MAX_FAISCEAU* **alors**
        $pire\_complexe := Argmin_{(co \in faisceau_2)} laplace(co, c)$
        $faisceau_2 := faisceau_2 - \{pire\_complexe\}$
  $faisceau_1 = faisceau_2$
retourner la règle : si $meilleur\_complexe$ alors classe=c

exemples E. Il retire de E tous les exemples de classe c couverts par la première règle et construit une deuxième règle à l'aide de l'ensemble $E - covers(H)$. Il itère ainsi jusqu'à ce que l'ensemble E ne possède plus aucun exemple de classe c. En pratique, l'ensemble d'exemples E peut au final ne pas être vide s'il n'est plus possible de construire de règles significatives, la condition de complétude est alors relâchée (voir définition 1.3). Cette stratégie est adoptée pour chacune des classes d'apprentissage.

1.1.3 Stratégies "séparer pour régner" et "diviser pour régner"

Les systèmes qui induisent des arbres de décision et le système CN2 sont représentatifs de différentes stratégies, respectivement celle dite *"diviser pour régner"* et celle *"séparer pour régner"*.

La première implique une partition de l'espace des exemples $\mathcal{E}_E$ et a fortiori de l'ensemble d'apprentissage E. En effet, pour un exemple donné, il existe un seul parcours depuis la racine jusqu'à une feuille de l'arbre ce qui signifie qu'un exemple est couvert exactement par une seule règle. La classification d'un exemple d'étiquette inconnue ne rencontre donc aucune difficulté : il n'y a pas de conflit entre règles de classes différentes couvrant un même exemple.

La seconde stratégie n'a pas cette contrainte. Lors de la construction d'une règle, celle-ci peut couvrir des exemples couverts par des règles préalablement construites. Un exemple peut être couvert par aucune règle ou au contraire par plusieurs règles. Les algorithmes de type *"séparer pour régner"* ont été étudiés par Fürnkranz (Fürnkranz,

1999), il propose un algorithme générique pour l'induction de type "séparer pour régner" et identifie deux types d'algorithmes, ceux qui ordonnent les règles apprises et ceux qui ne les ordonnent pas.

L'apprentissage de théories non ordonnées (algorithme 1.9) est celui mis en œuvre par le système CN2 tel que présenté dans la sous-section 1.1.2. Ainsi la méthode $construire\_une\_regle(E, c)$ est celle de l'algorithme 1.8 et le critère d'arrêt est l'induction d'une règle non significative. Les exemples retirés à chaque induction d'une règle r sont les exemples de classe courante couverts par r. Enfin, il n'y a pas d'élagage de la théorie ; l'élagage étant en général la suppression d'hypothèses.

Algorithme 1.9 : Algorithme de type "séparer pour régner" pour l'induction de théories non ordonnées

Entrées : - un ensemble d'exemples E
Sorties : - une théorie $HSet$
$HSet := \emptyset$
1. **pour chaque** *classe* c **faire**
  tant que E *contient des exemples de classe* c **faire**
    $H := construire\_une\_regle(E, c)$
    $cov := exs\_couverts(H, E, c)$
    si $critere\_arret(HSet, H, E)$ **alors**
      quitter la boucle **tant que**
    $E := E - cov$
    $HSet := HSet \cup \{H\}$
  $HSet := elagage(HSet)$
retourner $HSet$

L'apprentissage de théories ordonnées (Clark & Niblett, 1989; Frank & Witten, 1998), aussi appelées liste de décisions, est une variante de cet algorithme. La boucle extérieure (ligne 1) n'y est pas présente, c'est la méthode $construire\_une\_regle(E, c)$ qui fixe la classe de la règle induite ; par exemple c peut être fixé comme étant la classe majoritaire parmi les exemples couverts par la règle.

Mais la différence est surtout dans la façon d'interpréter les règles et de classer de nouveaux exemples, c'est à dire d'assigner une classe à un exemple dont on ne connaît pas a priori l'étiquette.

Pour une théorie non ordonnée, la classification d'un exemple se fait à l'aide de toutes les règles couvrant l'exemple. Le cas difficile est celui où plusieurs de ces règles sont de classes différentes. Différentes stratégies peuvent alors être adoptées pour décider de la classe de l'exemple (Ali & Pazzani, 1993; Michalski *et al.*, 1986a).

Pour une théorie ordonnée, l'exemple d'étiquette inconnue est confronté à chacune des règles dans l'ordre défini, la première règle qui couvre l'exemple lui assigne sa classe. Cette méthode de classification est simple mais rend l'interprétation de la théorie difficile ; en effet, l'interprétation d'une règle dépend des règles précédentes et devient impossible lorsque le nombre de règles est important. C'est pour cette raison que nous n'utilisons pas ce type d'algorithme.

Les deux types d'apprentissage "diviser pour régner" et "séparer pour régner" sont les principales approches adoptées en logique attribut-valeur mais ces concepts sont transposables à la recherche de programmes définis, c'est-à-dire lorsque que l'on s'appuie sur la logique des prédicats (voir annexe A).

1.2 Programmation logique inductive

Le langage utilisé par la programmation logique inductive (PLI) est le langage des prédicats (voir annexe A). Dans le cadre de l'apprentissage de programmes définis, ce langage exprime aussi bien les exemples ($\mathcal{L}_E$) que les hypothèses ($\mathcal{L}_H$). La relation de couverture *covers* est ici la conséquence logique ($\models$). Les intérêts majeurs de la PLI par rapport à d'autres techniques d'apprentissage sont :

- l'expressivité : celle ci est plus riche que dans la plupart des autres logiques. Un problème d'apprentissage en logique attribut-valeur peut être exprimé dans ce langage et l'inverse n'est pas toujours vrai ; en particulier, le langage des prédicats permet d'exprimer des relations entre objets.
- il est possible d'utiliser des connaissances a priori (en anglais *background knowledge*) que nous notons T.

Le revers de cette expressivité est une efficacité en général plus limitée lors de l'induction, entre autres expliquée par une taille d'espace de recherche plus importante et un test de couverture plus coûteux. Néanmoins, les outils de PLI sont utilisés dans de nombreuses applications (Lavrac & Dzeroski, 1993). Ces applications sont issues de domaines divers : la physique (Dolšak & Muggleton, 1992), l'écologie (Dzeroski *et al.*, 1999), la caractérisation des arythmies cardiaques (Fromont *et al.*, 2003; Quiniou *et al.*, 2001) ou encore la biologie (King *et al.*, 2004; Srinivasan *et al.*, 1996).

Dans cette section, nous allons tout d'abord présenter le problème d'apprentissage en PLI (sous-section 1.2.1), puis l'algorithme générique pour l'apprentissage de théories clausales proposé par Muggleton et De Raedt (Muggleton & De Raedt, 1994) est étudié dans la sous-section 1.2.2. La θ-subsomption, utilisée comme relation de généralité entre hypothèses, est une relation entre clauses permettant la structuration de l'espace des hypothèses (sous-section 1.2.3). Différents biais d'apprentissage (sous-section 1.2.4) et différentes stratégies de recherche (sous-section 1.2.5) sont introduits pour orienter et limiter les recherches. Enfin deux systèmes de PLI sont présentés : FOIL (sous-section 1.2.6.1) et Aleph (sous-section 1.2.6.2).

1.2.1 Sémantique

Les programmes de PLI s'imposent en général de respecter certaines contraintes connues sous le nom de *sémantique normale* (Muggleton & De Raedt, 1994). Elles correspondent à une reformulation de la cohérence d'une conjonction d'hypothèses avec les exemples. Nous nous restreignons ici à l'apprentissage supervisé de deux classes ($\oplus$ et $\ominus$). En sémantique normale, étant donné un ensemble d'exemples positifs $E^{\oplus}$, un ensemble d'exemples négatifs $E^{\ominus}$, une théorie des connaissances a priori T (*background*

knowledge), le but de la PLI est de construire une théorie $HSet$ (c'est-à-dire un ensemble de clauses) respectant les conditions de la sémantique normale.

Définition 1.10 (Sémantique normale).
Les ensembles d'exemples $E^{\oplus}$ et $E^{\ominus}$ sont ici représentés par les ensembles de littéraux positifs constituant les exemples et contre exemples[2]*. Posons $conj\_pos(E^{\oplus})$ la conjonction des littéraux de $E^{\oplus}$ et $conj\_neg(E^{\ominus})$ la conjonction des négations de littéraux de $E^{\ominus}$. Les conditions de la sémantique normale s'écrivent alors :*

- *Satisfaisabilité a priori : $T \wedge conj\_neg(E^{\ominus}) \not\models \square$ (faux)*
- *Satisfaisabilité a posteriori (correction) : $T \wedge HSet \wedge conj\_neg(E^{\ominus}) \not\models \square$*
- *Nécessité a priori : $T \not\models conj\_pos(E^{\oplus})$*
- *Suffisance a posteriori (complétude) : $T \wedge HSet \models conj\_pos(E^{\oplus})$*

Les apprentissages par PLI qui vérifient la sémantique normale sont nommés apprentissage par conséquence logique (de l'anglais *learning from entailment*). La première condition permet de s'assurer qu'aucun exemple négatif de $E^{\ominus}$ ne peut être déduit de la connaissance a priori T. La deuxième condition permet de s'assurer qu'aucun exemple négatif ne peut être déduit de la connaissance a priori et des hypothèses $HSet$ (voir définition 1.3). Il est nécessaire également que les exemples positifs $E^{\oplus}$ ne puissent pas être tous déduits de la connaissance a priori (3ème condition). Enfin, les connaissances a priori et les hypothèses induites doivent permettre de prouver tous les exemples positifs.

La *sémantique définie* est un cas particulier de la sémantique normale où les connaissances a priori et les hypothèses sont des ensembles de clauses définies tels que des programmes Prolog (exemple 1.12). Dans ce cas, il existe un unique plus petit modèle d'Herbrand $\mathcal{M}$ pour les programmes T, H et $H \wedge T$ dans lequel toute formule logique sans négation est, soit vraie, soit fausse.

Définition 1.11 (Sémantique définie).
Les conditions de la sémantique définie sont :

- *Satisfaisabilité a priori : $\forall e \in E^{\ominus}, e$ est faux dans $\mathcal{M}(T)$*
- *Satisfaisabilité a posteriori (correction) : $\forall e \in E^{\ominus}, e$ est faux dans $\mathcal{M}(T \wedge H)$*
- *Nécessité a priori : $\exists e \in E^{\oplus}$ tel que e est faux dans $\mathcal{M}(T)$*
- *Suffisance a posteriori (complétude) : $\forall e \in E^{\oplus}, e$ est vrai dans $\mathcal{M}(T \wedge H)$*

Exemple 1.12 (sémantique normale et définie).
On souhaite ici apprendre automatiquement la relation $mère(X, Y)$ (autrement dit le prédicat $mère/2$) qui se lit Y est la mère de X, à partir des prédicats $parent/2$ et $femme/1$. On dispose d'un ensemble d'exemples de classe $\oplus$ représentés sous forme de littéraux positifs, et d'un ensemble d'exemples négatifs :

$$E^{\oplus} = \{mère(anne, sophie), mère(mathieu, sophie)\}$$

$$conj\_pos(E^{\oplus}) = mère(anne, sophie) \wedge mère(mathieu, sophie)$$

[2]Nous considérons en effet que les exemples de $E^{\oplus}$ et $E^{\ominus}$ sont des littéraux positifs dans lesquels apparaissent les identifiants des exemples. La description des exemples, à base de clauses, est quant à elle donnée dans la théorie du domaine T.

$$E^{\ominus} = \{mère(sophie, mathieu), mère(mathieu, anne)\}$$

$$conj\_neg(E^{\ominus}) = \neg mère(sophie, mathieu) \wedge \neg mère(mathieu, anne)$$

On dispose également d'une théorie de connaissance a priori T constituée dans cet exemple de littéraux positifs uniquement :

$$T = femme(sophie) \wedge parent(anne, sophie) \wedge parent(mathieu, sophie)$$

Alors la théorie suivante, constituée d'une seule clause H respecte les quatre conditions des sémantique normale et définie :

$$H = \forall X \forall Y (mère(X, Y) \leftarrow parent(X, Y), femme(Y))$$

La majorité des systèmes de PLI sont basés sur la sémantique définie où les exemples sont des littéraux positifs instanciés. Il existe d'autres sémantiques telles que la *sémantique non monotone* (Muggleton & De Raedt, 1994) que je n'aborde pas ici.

1.2.2 Algorithme générique de PLI

S. Muggleton et L. De Raedt ont proposé un squelette pour les algorithmes de PLI (Muggleton & De Raedt, 1994).

Algorithme 1.13 : Squelette d'un algorithme de PLI

Entrées : - un ensemble d'exemples E
Sorties : - une théorie $HSet_{best}$
$QHSet$:= initialiser
tant que *le critère d'arrêt de $QHSet$ n'est pas rempli* **faire**
  choisir une conjonction d'hypothèse $HSet$ telle que $HSet \in QHSet$
  choisir les règles d'inférence $r_1, \ldots, r_k \in R$ à appliquer à $HSet$
  appliquer $r_1, \ldots, r_k$ à $HSet$ pour obtenir les théories $HSet_1, \ldots, HSet_n$
  ajouter $HSet_1, \ldots, HSet_n$ à $QHSet$
  élaguer $QHSet$
retourner la meilleure conjonction d'hypothèse $HSet_{best}$ de $QHSet$

L'algorithme construit de manière incrémentale un ensemble de théories $QHSet$. A chaque itération, il choisit une théorie $HSet$ de $QHSet$, génère les théories $HSet_1, \ldots,$ $HSet_n$ à l'aide de l'application des règles d'inférence de R puis met à jour $QHSet$ avec les nouvelles théories.

En théorie, les conditions de sémantique en PLI constituent un critère d'arrêt de l'algorithme 1.13. De même, le non respect des conditions de la sémantique par une théorie $HSet$ peut être une raison de l'élagage de l'ensemble des théories de $QHSet$, par la suppression dans $QHSet$ de $HSet$. En pratique, les conditions de la sémantique définie, en particulier la correction et la complétude (voir définition 1.11), sont relâchées. Il s'agit alors de construire une théorie qui permette la déduction du maximum d'exemples de $E^{\oplus}$ tout en s'autorisant la déduction d'un minimum d'exemples de $E^{\ominus}$.

En effet, le *bruit* dans les données, qui peut être constitué d'erreurs dans l'étiquetage des exemples, dans la description des exemples ou dans les connaissances a priori, contraint à relâcher ces contraintes si l'on veut généraliser les exemples.

On remarque ici que les algorithmes "séparer pour régner" et "diviser pour régner" de recherche de théories sont des cas particuliers de l'algorithme 1.13. Dans les deux cas, on ne maintient qu'une théorie dans $QHSet$. Les étapes d'élagage de $QHSet$ et de choix d'une théorie $HSet$ dans $QHSet$ n'ont donc pas lieu d'être. Dans le cas de la recherche "séparer pour régner", une règle d'inférence consiste en l'ajout d'une nouvelle clause à la théorie $HSet$ en se focalisant sur les exemples positifs non encore couverts. Dans le cas de la recherche "diviser pour régner", telle que la construction d'arbres de décision, une règle d'inférence remplace une clause de $HSet$ par deux autres ; ce qui est le résultat de l'ajout d'un nœud dans l'arbre de décision. La plupart des algorithmes de recherche de théories en logique des prédicats sont de type "séparer pour régner" ; cependant certains tels que TILDE (Blockeel & De Raedt, 1998) recherchent des arbres de décision dont les règles extraites sont exprimées en logique des prédicats.

Nous nous intéressons par la suite à la construction d'une clause (et non d'une conjonction de clauses) dans le cadre d'une recherche de théorie de type "séparer pour régner". Dans la sous-section 1.2.3, la structuration de l'espace des hypothèses est établie à l'aide de la définition d'une relation de généralité entre clauses. Différents types de biais d'apprentissages sont introduits dans la sous section 1.2.4 ; ceux ci permettent de limiter la taille de l'espace de recherche. Enfin les différentes stratégies de parcours de cet espace sont abordées dans la sous-section 1.2.5.

1.2.3 Structuration de l'espace de recherche de clauses

Dans une problématique d'apprentissage telle que définie par Mitchell (Mitchell, 1982), une relation de généralité ($\succeq$) entre clauses doit être définie pour structurer l'espace de recherche. Idéalement, dans le cadre des clauses, celle ci pourrait être définie à l'aide de la conséquence logique : c'est à dire, pour deux clauses définies C_1 et C_2, $C_1 \succeq C_2$ si et seulement si $C_1 \models C_2$.

Cependant, des résultats d'indécidabilité sur la relation de conséquence logique en empêche son utilisation en pratique même si l'on se restreint aux clauses de Horn (Nienhuys-Cheng & Wolf, 1996). En PLI, la relation de généralité la plus communément utilisée est la θ-subsomption (Plotkin, 1970) qui est décidable.

Définition 1.14 (θ-subsomption de deux clauses).
Une clause C_1 θ-subsume une clause C_2, que l'on note $C_1 \succeq_\theta C_2$, si et seulement si il existe une substitution θ telle que $C_1\theta \subseteq C_2$. En d'autres termes, il est possible de trouver une substitution θ telle que tous les littéraux de $C_1\theta$ [3] soient présents dans la clause C_2. On dit alors que C_1 est une généralisation de C_2 (ou que C_2 est une spécialisation de C_1) par θ-subsomption.

La θ-subsomption induit un ordre plus faible que la conséquence logique. En effet, $C_1 \succeq_\theta C_2$ implique que $C_1 \models C_2$ et donc que $covers(C_2)$ est inclus dans $covers(C_1)$, l'in-

[3] la clause $C_1\theta$ est la clause C_1 sur laquelle la substitution θ est appliquée (voir définition A.16).

verse n'est pas toujours vrai (Plotkin, 1971). Bien que la θ-subsomption soit décidable, contrairement à la conséquence logique, elle reste néanmoins NP-difficile (Kapur & Narendran, 1986). Il est cependant montré que les deux relations sont proches et mêmes équivalentes si on exclut l'utilisation de clauses récursives, c'est-à-dire les clauses qui ont un même symbole de prédicat dans la tête et dans le corps.

La θ-subsomption est une relation transitive entre clauses et permet la structuration de l'espace des hypothèses en un treillis. L'exemple 1.15 est un aperçu de la structuration de l'espace de recherche de clauses à l'aide de la cette relation.

Exemple 1.15 (structuration de l'espace de recherche par θ-subsomption). Dans cet exemple, on se pose le problème d'apprendre la relation $mère(X, Y)$, qui se lit Y est mère de X, à partir des relations $parent$ et $femme$. Les exemples, comme $mère(anne, sophie)$, et les contre-exemples sont des littéraux positifs. Les connaissances a priori T sont constituées des littéraux positifs définissant les relations $parent$ et $femme$, comme $parent(anne, sophie)$ et $femme(sophie)$.

La figure 1.16 représente une partie de l'espace des hypothèses $\mathcal{E}_H$ et certaines mises en relation par θ-subsomption de ces hypothèses.

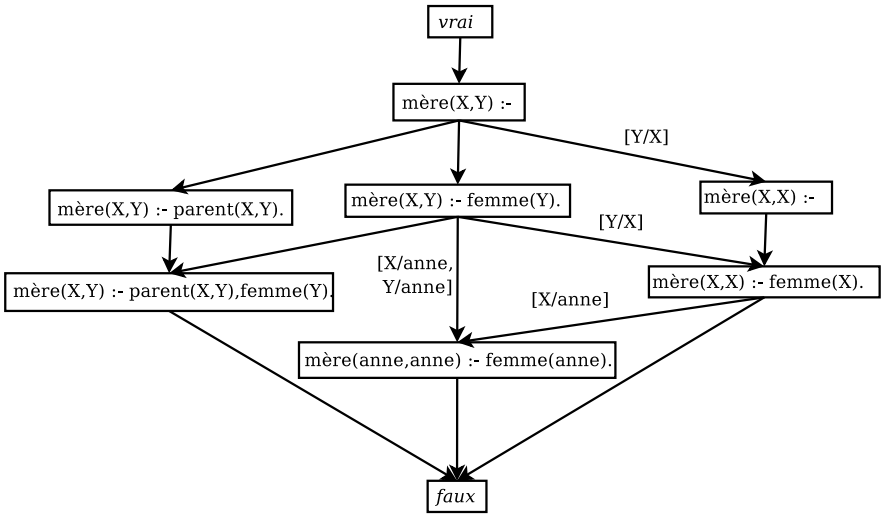


FIG. 1.16 – Structuration de l'espace de recherche de clauses sous θ-subsomption pour l'exemple 1.15. Les flèches représentent la relation de θ-subsomption, elles sont annotées par les substitutions utilisées si celles-ci ne sont pas vides. La clause $vraie$ θ-subsume toutes les clauses de l'espace, toute interprétation est modèle de $vrai$. La clause $faux$ est θ-subsumée par toutes les clauses et n'a aucun modèle.

1.2.4 Biais d'apprentissage

Pour Mitchell (Mitchell, 1980) un biais d'apprentissage est une méthode qui permet de sélectionner une hypothèse plutôt qu'une autre, en se basant sur des critères autres

que leur cohérence vis-à-vis des exemples d'apprentissage [4].

De nombreux types de biais existent (Nédellec *et al.*, 1996). Nous nous intéressons ici aux *biais déclaratifs* ainsi qu'aux *biais de préférence*. Les biais déclaratifs sont un moyen de restreindre l'espace de recherche ; en particulier, ils définissent les clauses dont la syntaxe est acceptable, par exemple pour un utilisateur. Les biais de préférence sont des heuristiques pour évaluer les clauses afin de les comparer entre elles.

1.2.4.1 Biais déclaratifs

Un biais déclaratif peut être une contrainte sur la forme des clauses et donc une restriction sur l'espace des hypothèses $\mathcal{E}_H$. Ainsi, on peut contraindre la taille maximale d'une clause (en nombre de littéraux), le nombre maximal de variables qu'elles possèdent ou bien encore les prédicats utilisés.

Ces biais sont parfois exprimés sous forme d'une grammaire telle que DLAB (Dehaspe & De Raedt, 1996) sur laquelle repose le système ICL (De Raedt & Van Laer, 1995). Cette grammaire définit en intention l'espace des hypothèses $\mathcal{E}_H$ (voir exemple 1.17).

Exemple 1.17 (le biais déclaratif DLAB**).**
Dans le formalisme DLAB, le corps d'une clause peut être exprimé par la grammaire G suivante : `2-2:[p(X,Y),0-2:[r(X),q(Y)]]` . Le membre `0-2:[r(X),q(Y)]` signifie que le corps de clause peut contenir entre 0 et 2 prédicats parmi les littéraux $r(X)$ et $q(Y)$. Les corps des clauses acceptables par la grammaire G est l'ensemble : $\{p(X,Y), p(X,Y) \wedge r(X), p(X,Y) \wedge q(Y), p(X,Y) \wedge r(X) \wedge q(Y)\}$.

Une forme de biais déclaratif très répandue (par exemple dans PROGOL (Muggleton, 1995) et dans son successeur Aleph) est la définition de modes et de types pour les variables internes d'un prédicat. Les modes permettent également de définir le type d'utilisation d'une variable (variable d'entrée ou de sortie de prédicat).

Exemple 1.18 (la déclaration de modes dans PROGOL et Aleph).
La déclaration du mode `mode(2,p(+t1,-t2,#t3))` spécifie qu'une clause peut avoir au plus deux occurrences du prédicat p/3 et que celles-ci doivent avoir :

- une variable de type $t1$, en premier argument, déjà apparue dans les littéraux précédant p dans la clause,
- une variable de type $t2$, en deuxième argument, non encore apparue dans les littéraux précédant p dans la clause,
- une constante de type $t3$ en troisième argument.

La variable de type de $t1$ est dite d'entrée alors que la variable de type $t2$ est dite de sortie.

[4] traduction de l'anglais : *"a bias refers to any basis for choosing one generalization over another, other than strict consistency with the observed training instances."*

1.2.4.2 Biais de préférence

L'utilisation d'un biais de préférence, aussi appelé heuristique de recherche, va diriger la recherche dans l'espace des hypothèses $\mathcal{E}_H$ en favorisant certaines clauses à d'autres. Il permet de mesurer l'adéquation d'une clause avec les exemples de la classe à prédire. Un biais de préférence sert également à éliminer des hypothèses ; par exemple une règle peut être éliminée si son évaluation ne dépasse pas un seuil préétabli. En général, pour évaluer une clause, l'heuristique de recherche s'appuie sur les exemples couverts par cette clause. Fürnkranz et Flach (Fürnkranz & Flach, 2003) proposent une revue et analyse des heuristiques utilisées en apprentissage de règles. L'heuristique *m-estimate* (Cestnik, 1990) est donnée ici dans l'exemple 1.19.

Exemple 1.19 (heuristique de recherche *m-estimate*).
On note n le nombre d'exemples d'apprentissage et $n^{\oplus}$ ceux qui sont de classe $\oplus$ parmi eux. Soit une clause C, on note $n(C)$ le nombre d'exemples couverts par C et $n^{\oplus}(C)$ le nombre d'exemples de classe $\oplus$ parmi eux. La probabilité a priori de la classe $\oplus$ est : $p_a(\oplus) = \frac{n^{\oplus}}{n}$. Le *m-estimate* est calculé de la manière suivante : $\frac{n^{\oplus}(C)+m*p_a(\oplus)}{n(C)+m}$ où m est un paramètre. Lorsque $m = 0$, le m-estimate est égal à la pureté de la règle (Fürnkranz, 1999) et lorsque $m = 2$ et $p_a(\oplus) = 0.5$, le m-estimate est égal à l'estimateur de Laplace (équation 1.7).

L'espace de recherche $\mathcal{E}_H$ peut donc être hiérarchisé à l'aide de la relation de θ-subsomption entre clauses, il est restreint à l'aide des biais déclaratif et de préférence et les hypothèses sont comparées entre elles à l'aide des biais de préférence. La sous-section suivante présente les différentes stratégies de parcours de l'espace de recherche.

1.2.5 Stratégie de parcours dans l'espace de recherche

Il existe plusieurs stratégies de recherche dont les recherches descendante, ascendante ou mixte qui sont présentées ci-dessous. Elles sont basées en général sur des opérateurs dits de *raffinement* (Shapiro, 1983). Un opérateur d'inférence déductive (resp. inductive) construit une clause C' à partir d'une clause C telle que C' est plus spécifique que C (respectivement C' est plus générale que C). Dans le cadre d'un espace de recherche structuré par la relation de θ-subsomption, un opérateur d'inférence déductive (resp. inductive) construit donc une clause C' à partir d'une clause C telle que C θ-subsume C' (resp. C' θ-subsume C). Trois stratégies de parcours sont présentées ici, elles s'appuient sur la définition de tels opérateurs. On note que les choix de stratégie et de biais sont très liés.

1.2.5.1 Recherche descendante

La recherche descendante (en anglais *top-down*) consiste à considérer tout d'abord la clause la plus générale du treillis, c'est-à-dire la clause *vrai* [5].

[5] En pratique , la clause utilisée est la clause la plus générale du langage défini par le biais déclaratif, cette clause est souvent plus spécifique que *vrai* mais reste néanmoins plus générale que toute clause du langage défini par le biais.

L'algorithme de recherche descendante consiste à appliquer, successivement, aux clauses considérées, des opérateurs d'inférence déductive. Lorsque l'espace des clauses est hiérarchisé par la relation de θ-subsomption, les opérateurs d'inférence déductive classiques sont :

- l'ajout de littéral à la clause
- l'application d'une substitution, afin d'unifier différentes variables ou remplacer une variable par une constante

Un biais déclaratif peut contraindre par exemple à rassembler en une seule étape plusieurs de ces opérateurs.

Ce type de recherche est typiquement basé sur des algorithmes *générer et tester* (que l'on oppose aux algorithmes *guidés par les exemples* - voir sous-section 1.2.5.2). Ces algorithmes ont deux étapes distinctes : la génération d'une clause à partir d'une clause plus générale et d'opérateurs d'inférence déductive et le test (évaluation) de cette clause sur les exemples.

Dans le cadre de la sémantique définie, l'algorithme FOIL (Quinlan, 1990; Quinlan & Cameron-Jones, 1993) est basé sur cette stratégie. CLAUDIEN (De Raedt & Dehaspe, 1997), ICL (De Raedt & Van Laer, 1995) et TILDE (Blockeel & De Raedt, 1998) sont des algorithmes par recherche descendante dans le cadre d'une sémantique non monotone.

1.2.5.2 Recherche ascendante

La recherche ascendante (en anglais *bottom up*) consiste à partir d'une hypothèse spécifique de l'espace de recherche et lui appliquer une succession d'opérations d'inférence inductive. En général, la génération de clause dépend d'un ou plusieurs exemples ; les exemples sont en effet les bornes les plus spécifiques de l'espace de recherche et constituent le point de départ d'une recherche ascendante. Les algorithmes de recherche ascendante sont dits *guidés par les exemples*.

Une première approche consiste à construire la clause *lgg* (acronyme de l'anglais *least general generalization*) de deux exemples d'apprentissages exprimés sous forme de clauses (Plotkin, 1970). Intuitivement, la *lgg* de deux clauses est leur généralisation commune la moins générale. Elle est le résultat d'une opération d'inférence inductive qui généralise le moins les deux exemples. Les détails sur la construction de la lgg de deux clauses sont donnés dans (Cornuéjols & Miclet, 2002) et (Lavrac & Dzeroski, 1993). Nous nous contenterons ici de donner un exemple de lgg (exemple 1.20).

Exemple 1.20 (construction d'une *lgg*).
Soient C_1 et C_2 deux clauses :
$C_1 = mère(anne, sophie) : -parent(anne, sophie), femme(sophie), femme(anne).$
$C_2 = mère(franck, marie) : -parent(franck, marie), femme(marie), homme(franck).$
La lgg de C_1 et C_2 est alors :
$lgg(C_1, C_2) = mère(X, Y) : -parent(X, Y), femme(Y).$

La lgg ne s'appuie que sur les exemples d'apprentissage et afin de prendre en compte

la théorie du domaine T, Plotkin propose la *rlgg* (acronyme de l'anglais *relative least general generalization*) de deux clauses C_1 et C_2 relativement aux connaissances a priori T qu'il définit de la manière suivante :
$rlgg(C_1, C_2) = lgg(C_1 \leftarrow \mathcal{M}(T), C_2 \leftarrow \mathcal{M}(T))$
Où $\mathcal{M}(T)$ est le plus petit modèle d'Herbrand de T. La rlgg est utilisée par l'algorithme GOLEM (Muggleton & Feng, 1990) et ses successeurs. Mais son utilisation en pratique est rendue difficile par sa taille. En effet, la taille de $\mathcal{M}(T)$, et par construction celle de la rlgg, peut être infinie. De plus, elle croît exponentiellement avec le nombre d'exemples.

Une autre approche de recherche ascendante consiste à inverser le mécanisme de déduction de Prolog, c'est à la dire la résolution SLD (voir annexe A). Cette approche (nommée en anglais *inverting resolution*), introduit deux types d'opérateur d'inférence inductive, les opérateurs V (opérateurs d'*absorption* et d'*identification*) et les opérateurs W (opérateurs d'*intra construction* et d'*inter construction*). Les opérateurs V inversent une étape de la résolution SLD et les opérateurs W résultent de la combinaison de deux opérateurs V. Ces derniers présentent la particularité d'introduire un nouveau prédicat qui n'apparaissait pas dans les pré-conditions des règles. Cornuéjols et Miclet (Cornuéjols & Miclet, 2002) présentent ces différents opérateurs.

Exemple 1.21 (l'opérateur d'inférence inductive V d' absorption).
L'opérateur V d'*absorption* repose sur le fait qu'à partir des deux clauses[6] $q \leftarrow A$ et $p \leftarrow q, B$ on peut déduire $q \leftarrow A$ et $p \leftarrow A, B$. A partir de ces deux dernières clauses, on induit alors $p \leftarrow q, B$. Cet opérateur est présenté dans la figure 1.22 ainsi qu'un exemple de son utilisation pour l'inférence. Dans cet exemple la clause c est induite à partir d'un exemple e et d'une clause de la théorie du domaine b [7].

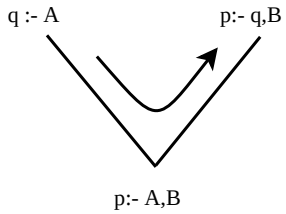


FIG. 1.22 – L'opérateur V d'absorption

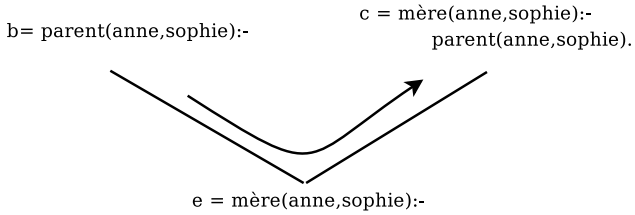


FIG. 1.23 – Une étape d'inférence inductive avec l'opérateur V d'absorption

De nombreux systèmes de PLI s'appuient sur ces opérateurs V et W d'inférence inductive dont l'un des premier est CIGOL (Muggleton & Buntine, 1988).

1.2.5.3 Recherche mixte

La recherche mixte est, au sein d'un seul système, la combinaison des stratégies de recherche ascendante et descendante. Dans cette sous section, nous nous attardons

[6] Dans cette notation q et p sont des littéraux positifs, A et B sont des conjonctions de littéraux positifs, éventuellement vides.

[7] Dans cet exemple, on remarque que $b \wedge c \models e$ et pourtant $e \not\models c$ (en effet, e θ-subsume c)

uniquement sur l'approche employée au sein du système PROGOL (Muggleton, 1995) et de son successeur Aleph. Cette approche est certainement la plus représentative de la stratégie de recherche mixte.

Dans cette approche, l'inférence inductive se base sur l'inversion de l'implication (en anglais *inverse entailment*). En considérant la recherche d'une seule clause définie h couvrant au moins un exemple e, la condition sémantique de suffisance a posteriori (voir la sous-section 1.2.1) peut être réécrite en $T \wedge h \models e$. Cette relation est équivalente à $T \wedge \bar{e} \models \bar{h}$ où $\bar{x}$ est la négation de x. Puisque h et e sont des clauses, leur négation sont donc des programmes constitués de clauses unitaires (c'est-à-dire avec un seul littéral). Soit $\bar{\bot}$ la conjonction des littéraux qui sont vrais dans tous les modèles de $T \wedge \bar{e}$, puisque tout modèle de $T \wedge \bar{e}$ doit être modèle de $\bar{h}$ alors on a la relation $T \wedge \bar{e} \models \bar{\bot} \models \bar{h}$. Et donc finalement $h \models \bot$.

Dans les systèmes PROGOL et Aleph, la recherche d'une clause se passe en deux étapes. La première étape est la construction de $\bot$ (appelée en anglais *bottom clause*) pour un exemple[8], c'est une étape de recherche ascendante (voir exemple 1.24). Cette clause constitue une base de littéraux pour la deuxième étape qui est une étape de recherche descendante. La seconde étape constitue la recherche d'un sous ensemble de ces prédicats qui forment la clause, elle s'appuie sur un algorithme A^* (Nilsson, 1980).

Exemple 1.24 (clause $\bot$ pour l'inversion de l'implication).
Soit un exemple $e = mère(anne, sophie)$, une théorie du domaine T constituée des clauses unitaires $parent(anne, sophie)$ et $femme(sophie)$. $\bar{\bot}$ est la conjonction des littéraux vrais dans tout modèle de $T \wedge \bar{e}$, c'est-à-dire :
$\bar{\bot} = \neg mère(anne, sophie) \wedge parent(anne, sophie) \wedge femme(sophie)$. Finalement :
$\bot = mère(anne, sophie) \leftarrow parent(anne, sophie) \wedge femme(sophie)$.

En comparaison de l'inversion de la résolution (voir sous-section 1.2.5.2) qui consiste en une succession d'étapes d'inférence inductive, l'inversion de l'implication (utilisée pendant la phase de recherche ascendante) se déroule en une seule étape déterministe, ce n'est donc pas une recherche à proprement parler.

1.2.5.4 Élagage et limitation de la recherche

Malgré la définition de biais déclaratifs (voir sous-section 1.2.4.1), l'espace des hypothèses est très souvent trop important pour être parcouru entièrement. Les systèmes utilisent alors des méthodes pour élaguer l'espace des hypothèses et limiter la recherche. Ces méthodes dépendent de la stratégie de recherche et des biais employés.

Par exemple, dans une recherche descendante, si on construit une clause C qui ne couvre aucun exemple de $E^{\oplus}$, toute autre clause construite à partir de C, donc moins générale que C, ne couvre aussi aucun exemple positif. De cette manière on sait que l'espace des clauses moins générales que C peut être élagué.

Afin de limiter l'espace de recherche parcouru, une *recherche par faisceau* (celle utilisée par le système CN2, sous-section 1.1.2) peut être mise en œuvre. Cela consiste

[8] $\bot$ est potentiellement de cardinalité infinie, PROGOL et Aleph utilisent alors la déclaration de modes (voir sous-section 1.2.4.1) pour restreindre sa taille.

à maintenir l'ensemble, nommé faisceau, des clauses déjà construites pour lesquelles l'heuristique de recherche est maximisée (c'est-à-dire les clauses qui discriminent le mieux les exemples $E^{\oplus}$ des exemples $E^{\ominus}$). La taille du faisceau est un paramètre et les opérateurs d'inférence ne sont appliqués qu'aux clauses du faisceau, les autres étant ignorées.

1.2.6 Deux systèmes de PLI

Dans cette sous section, deux systèmes de PLI sont présentés : FOIL et Aleph. Ces deux systèmes s'appuient sur une sémantique normale et leur algorithme de construction de théorie (ensemble de clauses) adopte, au moins au début, des stratégies de type "séparer pour régner" (voir sous-section 1.1.3). FOIL est un système de recherche descendante alors que Aleph est un système de recherche mixte.

1.2.6.1 FOIL : un système de recherche descendante

Le but du système FOIL (Quinlan, 1990; Quinlan & Cameron-Jones, 1993) est de construire un classifieur $HSet$, sous la forme d'un programme défini couvrant tous les exemples positifs $E^{\oplus}$ et aucun exemple $E^{\ominus}$. L'algorithme général de FOIL est un algorithme de type "séparer pour régner", le concept à apprendre est le prédicat $c(X_1, \ldots, X_n)$.

Algorithme 1.25 : Algorithme de FOIL

Entrées : - un ensemble d'exemples $E = E^{\oplus} \cup E^{\ominus}$
Sorties : - un programme défini $HSet$
$HSet := \emptyset$
$Pos := E^{\oplus}$
tant que $Pos \neq \emptyset$ **faire**
  $Neg := E^{\ominus}$
  $C := c(X_1, \ldots, X_n) \leftarrow$
  tant que $Neg \neq \emptyset$ **faire**
    Construire un littéral L à ajouter à C
    Retirer de Neg les exemples négatifs non couverts par C
  Ajouter au programme $HSet$ la clause C
  Retirer de Pos les exemples positifs couverts par C
retourner le programme défini $HSet$

Lors de la recherche d'une hypothèse pour la classe $\oplus$, l'algorithme de FOIL (Algorithme 1.25) s'intéresse à l'ajout de deux types de littéraux, ceux qui permettent l'exclusion d'exemples négatifs - il utilise pour cela le calcul du *gain d'information* d'un littéral - et ceux qui permettent l'ajout de nouvelles variables.

Le *gain d'information* est une heuristique de recherche (voir sous-section 1.2.4.2). Soit L un littéral, C_i la clause en construction dans l'algorithme et C_{i+1} la clause C_i à laquelle est ajouté L. Pour une clause C_j, FOIL gère deux ensembles d'instances $T_j^{\oplus}$

et $T_j^{\ominus}$. Un élément de $T_j^{\oplus}$ (respectivement $T_j^{\ominus}$) est une instance liée[9] de la clause C_j correspondant à un exemple positif (respectivement négatif). Le calcul du *gain* dépend des clauses C_i, C_{i+1} et du littéral L :

$$gain(C_i, C_{i+1}, L) = n^{\oplus}(C_i) * \left(log_2(\frac{||T_{i+1}^{\oplus}||}{||T_{i+1}^{\oplus}|| + ||T_{i+1}^{\ominus}||}) - log_2(\frac{||T_i^{\oplus}||}{||T_i^{\oplus}|| + ||T_i^{\ominus}||})\right)$$

Où $n^{\oplus}(C_i)$ est le nombre d'exemples positifs couverts par C_i et $||T_j||$ est le cardinal de l'ensemble T_j.

Du fait de sa méthode d'apprentissage, FOIL permet de traiter les données bruitées. En effet, il peut considérer qu'une certaine partie des données est bruitée et arrêter la construction de la clause alors que des exemples négatifs sont encore couverts.

Une des difficultés de FOIL est un problème d'efficacité lié au calcul coûteux des ensembles $T_j^{\oplus}$ et $T_j^{\ominus}$. Une autre difficulté, cette fois ci pour l'utilisateur, est la nécessaire représentation de la connaissance a priori T en extension ; la connaissance a priori ne doit contenir que des littéraux totalement instanciés.

De nombreux systèmes de PLI se sont inspirés de FOIL. Par exemple FOCL (Pazzani & Kibler, 1992) est une extension de FOIL permettant la prise en compte d'une connaissance a priori définie en intention, pouvant contenir des clauses non unitaires. FOIDL (Mooney & Califf, 1995) permet aussi la définition d'une connaissance a priori en intention et construit des listes de décision (les règles sont ordonnées). MFOIL (Džeroski, 1993) propose de remplacer le gain d'information par l'heuristique de recherche m-estimate (voir exemple 1.19).

1.2.6.2 Aleph : un système de recherche mixte

Aleph (Srinivasan, 2003) est un successeur de PROGOL (Muggleton, 1995). A l'origine, Aleph (Algorithme 1.26) est un algorithme de type "séparer pour régner" se basant sur la recherche mixte telle que décrite dans la section 1.2.5.3. L'étape de saturation et de réduction sont respectivement les étapes de recherche ascendante et descendante de la recherche mixte. La déclaration de modes est un biais déclaratif (voir sous-section 1.2.4.1) utilisé pour la construction de $\perp$ et pour la restriction de l'espace des hypothèses dans l'étape de recherche ascendante.

L'algorithme est hautement paramétrable, Aleph a en effet été conçu pour proposer dans un outil unique, la mise en œuvre de différents biais, stratégies et élagages proposés dans divers systèmes tels que PROGOL, FOIL (Quinlan & Cameron-Jones, 1993), FORS (Karalič & Bratko, 1997), TILDE (Blockeel & De Raedt, 1998), MIDOS (Wrobel, 1997) et WARMR (Dehaspe, 1999). Par exemple, un ensemble d'heuristiques de recherche est disponible pour l'évaluation des clauses mais il est possible de définir ses propres heuristiques.

[9] Une instance liée de C_j correspondant à un exemple e peut être vue comme une substitution des variables de C_j faisant correspondre à chacune des variables une constante, et qui est le résultat d'une déduction de l'exemple e à l'aide de la clause C_j. Pour une clause et un exemple donnés, il peut donc y avoir plusieurs instances liées.

Algorithme 1.26 : Algorithme d'Aleph

Entrées : - un ensemble d'exemples $E = E^{\oplus} \cup E^{\ominus}$
Sorties : - un programme défini $HSet$
$HSet := \emptyset$
$Pos := E^{\oplus}$
tant que $Pos \neq \emptyset$ **faire**
1. Choisir aléatoirement e parmi $E^{\oplus}$
2. **Étape de saturation** : construire la clause $\perp$ la plus spécifique de l'espace de recherche telle que $T \wedge e \models \perp$. Cette clause, aussi appelée *bottom clause*, est généralement une clause définie contenant un grand nombre de littéraux
3. **Étape de réduction** : parcourir l'espace de recherche des clauses C plus générales que $\perp$, telles que tout littéral de la formule C est présent dans la formule $\perp$ ($C \subseteq \perp$)
4. Soit C_b la clause au meilleur score :
– Ajouter C_b au programme défini $HSet$
– Enlever de Pos les exemples positifs couverts par C_b

retourner le programme défini $HSet$

Enfin, Aleph propose à l'utilisateur d'établir l'opérateur d'inférence déductive utilisé lors de la recherche descendante (l'étape de réduction). L'utilisateur doit alors définir le prédicat $refine(C_i, C_{i+1})$ où C_i et C_{i+1} sont des clauses. Cet opérateur détermine la construction de nouvelles clauses C_{i+1} à partir d'une clause C_i. Pour effectuer une recherche descendante, il faut tout de même s'assurer qu'il respecte le fait que la clause C_i θ-subsume la clause C_{i+1}. Par contre, il peut être la concaténation de plusieurs opérateurs d'inférence déductive élémentaires tels que l'ajout d'un prédicat à la clause où l'application d'une substitution à C_i. Aleph teste alors sur les exemples toutes les clauses C_{i+i} construites par le prédicat $refine/2$ qui sont incluses dans $\perp$. On peut même décider de ne pas construire la clause $\perp$, Aleph ne vérifie alors pas l'inclusion de C_{i+1} dans $\perp$. Cette dernière approche revient finalement à une recherche de clauses exclusivement descendante (comme FOIL par exemple) où l'opérateur d'inférence déductive est définie par l'utilisateur.

1.3 Conclusion

Différentes techniques d'apprentissage de règles de classification sont exposées dans ce chapitre.

Dans le cadre d'une logique attribut-valeur le système CN2 et l'apprentissage d'arbres de décision sont présentés. Le système CN2 permet l'induction, pour chacune des classes d'apprentissage d'un ensemble de règles de classification. La classification de nouveaux exemples se fait alors à l'aide de l'ensemble des règles. Les arbres de décisions sont des structures permettant la classification de nouveaux exemples en parcourant la structure d'arbre induite de haut en bas ; c'est-à-dire depuis la racine de l'arbre jusqu'à ses feuilles.

Les noeuds d'un arbre représentent des attributs de l'application, exceptés les feuilles qui représentent des classes d'apprentissage. En utilisant un langage attribut-valeur, on fait l'hypothèse que l'application peut être décrite uniquement par un ensemble d'attributs.

La programmation logique inductive (PLI) recouvre l'ensemble des méthodes permettant l'induction de règles dans le cadre du langage des prédicats. Ce langage, plus expressif que le langage attribut-valeur, permet en particulier de décrire des relations entre objets de l'application. Le revers de cette expressivité est, en général, un coût plus important, en terme de temps et d'espace mémoire pour la génération des règles. Les spécifications de l'espace de recherche et de la stratégie adoptée pour son parcours sont donc décisives. Deux systèmes de PLI sont présentés ici, FOIL et Aleph. Le premier adopte une stratégie de recherche descendante dans l'espace des hypothèses. Le second adopte une stratégie mixte (ascendante puis descendante) ; on note toutefois qu'Aleph peut adopter une stratégie de recherche exclusivement descendante.

On différencie enfin deux types de systèmes, ceux qui utilisent une stratégie de type "diviser pour régner" et "séparer pour régner". On s'intéresse ici à la différence, au vu des ensembles de règles générés, dans l'adéquation de ces règles avec les exemples. Dans le premier cas, un exemple est couvert exactement par une seule règle. Dans le deuxième cas, cette contrainte n'existe pas ; ainsi, plusieurs règles peuvent être différentes explications d'un même exemple.

Chapitre 2

Le modèle SACADEAU et la simulation de scénarios

Le projet SACADEAU (Système d'Acquisition de Connaissances pour l'Aide à la Décision pour la qualité de l'EAU) vise à fournir une démarche et des outils permettant aux gestionnaires d'un territoire d'améliorer les pratiques agricoles et les aménagements du territoire dans un objectif de maîtrise de la qualité de l'eau. Il s'est focalisé sur une application portant sur le désherbage du maïs et son impact sur la contamination des eaux superficielles dans un contexte de systèmes agricoles liés à l'élevage, où la part du maïs dans l'assolement est importante. Le site d'application est celui du bassin versant du Frémeur dans le Morbihan.

Dans une première étape, le projet s'est consacré à l'élaboration d'un modèle de simulation des pratiques de désherbage des cultures et du transfert d'herbicides qui en résulte, afin d'en évaluer les impacts en terme de qualité de l'eau (Tortrat, 2005). Ce simulateur permet ainsi de tester différents *scénarios* afin d'évaluer l'impact des stratégies de désherbage, de la variabilité climatique et des aménagements ou les configurations spatiales du territoire.

Les résultats de simulation sont utilisés, dans une seconde étape du projet, pour découvrir, par des techniques d'apprentissage symbolique, les variables explicatives des phénomènes observés ou simulés et pour établir les relations entre ces variables afin d'avoir une meilleure compréhension et visibilité des facteurs importants. Ces informations permettent de produire des représentations médiatrices adaptées à la décision telles que des cartes, à l'échelle du bassin versant, illustrant les risques de transfert d'herbicide.

La section 2.1 introduit, en se focalisant sur le modèle développé, des notions sur le transfert d'eau et de pesticides à travers un bassin versant, ainsi que sur les modes de décision des agriculteurs pour le désherbage du maïs. La section 2.2 constitue un bilan des entrées et sorties du modèle. Nous nous intéressons particulièrement à la représentation des données, un découpage du bassin versant en sous-bassins appelés *arbres d'exutoires* est proposé. Un arbre d'exutoire est en général le territoire drainé par un point de sortie vers le réseau hydrographique, il peut correspondre à plusieurs parties

de parcelles : celle contiguë au réseau hydrographique, notée P, ainsi que celles en amont de P. Des variables de sorties sont définies aux différentes échelles considérées, qui sont le bassin versant et l'arbre d'exutoire, afin de caractériser le transfert d'herbicides. Dans la section 2.3, les conclusions d'une étude sur l'impact de plusieurs facteurs, réalisée à partir du modèle, sont données. Cette étude s'appuie sur la simulation de scénarios.

2.1 Modélisation des processus

Deux modèles interagissent pour assurer la simulation de tous les processus pris en compte dans ce projet. L'un représente les processus biophysiques (sous-sections 2.1.2 et 2.1.3) et l'autre les processus de décision des agriculteurs (sous-section 2.1.4). Avant de présenter le premier, nous introduisons des notions sur les écoulements au sein d'un bassin versant (sous-section 2.1.1). Un bassin versant représente le territoire drainé par une rivière en amont d'un point nommé l'exutoire (figure 2.1).

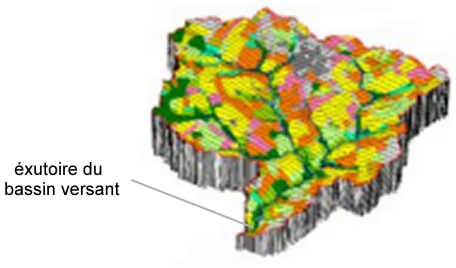


FIG. 2.1 – Le bassin versant du Frémeur d'une superficie de 17 km^2.

2.1.1 Modélisation des processus hydrologiques

Plusieurs processus d'écoulements interviennent lors d'un transfert d'eau et d'éléments chimiques associés dans un bassin versant (figure 2.2).

L'*infiltration* et la *recharge de nappe* sont les processus correspondant aux quantités entrantes et s'écoulant dans le sol jusque la nappe. Des modèles dits à base physique s'appuient généralement sur l'équation de Richards (Richards, 1931) pour simuler cet écoulement vertical au cours du temps. Cette équation permet le calcul du flux volumique d'eau vertical dans le sol en considérant le potentiel de l'eau et la teneur en eau à un point donné dans le sol, ceux-ci reposant sur les forces de gravité et les potentiels de pression tels que l'effet de capillarité. TOPMODEL (Beven & Kirkby, 1979) est quant à lui un modèle conceptuel basé sur un système à réservoir. Tant qu'un réservoir n'est pas rempli, l'eau ne pénètre pas dans un réservoir plus profond. D'autre modèles, dits empiriques, s'appuient sur la notion de décroissance de l'infiltration ; une formule de décroissance souvent utilisée est celle de Horton (Horton, 1938). La capacité d'infiltration, qui est le flux maximal entrant dans le sol détermine le partage entre l'infiltration et le

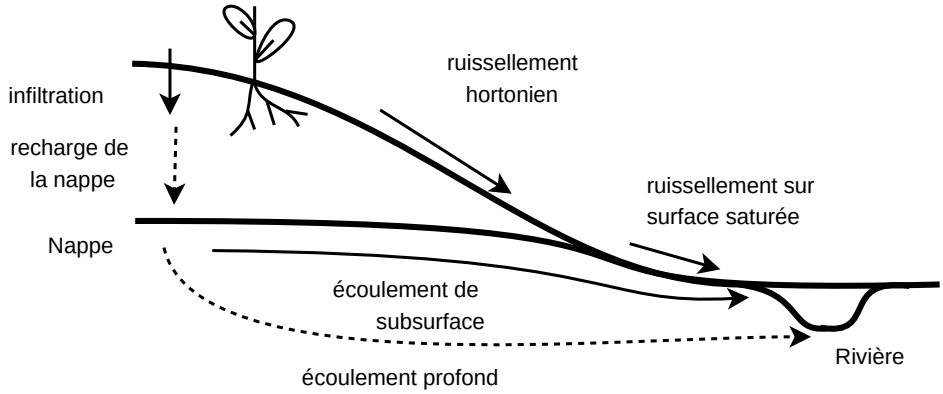


FIG. 2.2 – Les différents écoulements dans un bassin versant. L'écoulement profond de nappe et la recharge hydrique de la nappe ne sont pas pris en compte dans le modèle SACADEAU. La recharge de la nappe en pesticides par contre est prise en compte.

ruissellement. Elle est en effet importante après un travail du sol (tel qu'un labour) et décroît au cours du temps du fait de l'évolution de la structure du sol, la rugosité du sol se lissant et sa porosité se fermant.

Le *ruissellement hortonien* se produit lorsqu'une pluie dépasse la capacité d'infiltration du sol. Le modèle STREAM (Cerdan *et al.*, 2001) s'appuie sur des tableaux experts, pour des sols limoneux, afin de déterminer la capacité d'infiltration et donc le ruissellement hortonien en fonction du couvert végétal et de l'état de surface du sol (comprenant la rugosité et l'état structural du sol).

Le *ruissellement sur surface saturée* se produit au niveau des zones où la nappe affleure, où il n'y a pas d'infiltration possible. La modélisation de cet écoulement nécessite de déterminer ces zones. Au sein du modèle TOPMODEL (Beven & Kirkby, 1979), les surfaces saturées sont localisées en fonction des déficits de saturation locaux, c'est-à-dire de la quantité d'eau qu'il est nécessaire d'apporter pour saturer le sol en eau. Le calcul de ces déficits s'appuient sur le débit observé à l'exutoire du bassin versant et l'indice topographique de Beven (Beven & Kirkby, 1979). Pour un point P donné du bassin versant, cet indice dépend de l'aire drainée $draine(P)$ (c'est-à-dire la surface amont en hectare s'écoulant jusqu'à P) et de la pente locale au point P (voir figure 2.6) :

$$beven(P) = ln(\frac{draine(P)}{tan(\beta_P)}) \tag{2.3}$$

Un indice de Beven important signifie que la nappe est mal drainée, et donc souvent proche du sol, P se trouve alors probablement en fond de bassin versant.

Pour la modélisation des *écoulements nappe* (profond et de subsurface), des modèles conceptuels sont basés sur un découpage du profil de sol en colonnes, la profondeur du toit de la nappe étant propre à chaque colonne.

Le *transfert de pesticides* sur la surface du sol peut se faire sous forme dissoute dans l'eau de ruissellement ou sous forme particulaire adsorbée (c'est-à-dire fixée) sur des

matières en suspension. Ce deuxième processus est lié à l'érosion du sol, peu importante pour les pesticides et négligé ici. La modélisation des transferts de pesticides dans le sol est difficile, car la porosité joue un rôle important. Des voies préférentielles, telles que celles liées à la macroporosité du sol, sont les plus représentées, elles impliquent un transfert rapide et important.

Les choix effectués pour le développement du modèle de transfert SACADEAU reposent sur une approche conceptuelle et des variables faciles à observer. L'objectif est de pouvoir simuler les transferts à l'aide de variables internes observables et interprétables. Les partis pris du modèle sont les suivants :

- le partage ruissellement-infiltration s'appuie sur des tableaux experts en fonction de l'état structural de surface du sol, comme (Cerdan *et al.*, 2001) ;
- la localisation des zones saturées et la modélisation de subsurface s'appuie sur un découpage du profil de sol en colonnes dont les profondeurs de nappe sont calculées dynamiquement avec le débit et l'indice topographique de Beven (équation 2.3) par assimilation des gradients hydrauliques aux gradients topographiques.
- les écoulements profonds de nappe ne sont pas modélisés car lents ;
- la recharge de la nappe, difficile à modéliser, n'est pas prise en compte pour l'eau ; la recharge à la nappe des pesticides est estimée par un abattement des quantités mobilisées par l'eau d'infiltration, en fonction de la profondeur de nappe.

À ce modèle, représentant les processus biophysiques de transfert, est couplé un modèle des processus de décision des agriculteurs pour le désherbage du maïs.

2.1.2 Modélisation spatiale du bassin versant

La modélisation des écoulements dans SACADEAU s'appuie sur des spatialisations différentes de la surface du sol et du bassin versant. L'une est liée au ruissellement, l'autre aux écoulements de subsurface.

2.1.2.1 Ruissellement

La modélisation des processus biophysiques part de l'hypothèse que les flux (d'eau et d'herbicides) en sortie de bassins versants sont principalement contrôlés par la connectivité des écoulements liée à la mosaïque paysagère (Tortrat, 2005). En effet les flux sortants sont très faibles par rapport aux flux entrants : ils sont liés aux flux entrants mais aussi aux configurations spatiales (allocation spatiale des applications et aménagement de l'espace). Le modèle s'appuie donc sur une description détaillée des cheminements de l'eau dans le bassin versant.

Pour le ruissellement (Tortrat *et al.*, 2004), il faut déterminer l'ensemble des chemins empruntés par l'eau et les herbicides. Les flux sont évacués d'une parcelle en un point d'évacuation appelé exutoire de parcelle. Par abus de langage, nous assimilerons par la suite un exutoire d'une parcelle p à l'aire drainée de p par un de ses points d'évacuation.

Définition 2.4 (Exutoire de parcelle).
Un exutoire e d'une parcelle p est l'aire, contenue dans p, drainée par un point d'évacua-

tion de p. En particulier, il n'existe pas d'échange de flux entre e et tout autre exutoire constituant p.

Afin de déterminer les exutoires et les relations entre eux (figure 2.5) une carte d'occupation du sol et un modèle numérique d'altitude (MNA) du bassin versant sont utilisés. La carte d'occupation du sol indique les limites de parcelles ainsi que leur utilisation. Le MNA est une base de données donnant l'altitude d'un nœud d'une grille régulière, ici de 20 mètres. La détermination des exutoires et des relations se déroule en trois étapes :

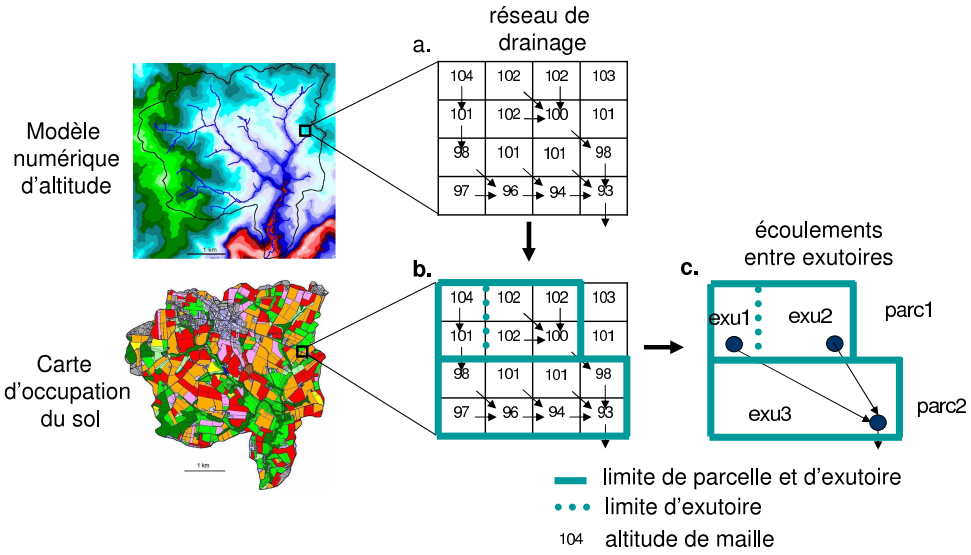


FIG. 2.5 – Processus en 3 étapes de détermination des écoulements entre exutoires de parcelles ; **a** : détermination du réseau de drainage. **b** : superposition du réseau et de la carte d'occupation du sol. **c** : détermination des exutoires et des écoulements entre exutoires. Les exutoires de la parcelle $parc1$, par exemple, sont $exu1$ et $exu2$.

a. Le réseau de drainage est construit à partir d'un schéma mono-directionnel des écoulements[1]. L'eau ruissellera d'une maille à la maille voisine qui a la plus faible altitude c'est à dire qui se trouve dans la pente la plus importante. Des fonctionnalités peuvent être attachées aux frontières de maille ; par exemple, une haie, au niveau d'une limite de maille, est vue comme un obstacle pour le ruissellement et implique la modification du réseau de drainage afin de prendre en compte cet obstacle.

b. La carte d'occupation du sol et le réseau de drainage sont superposés, en faisant correspondre au mieux les frontières de parcelle et les frontières des mailles.

[1]On parle de modélisation mono-directionnelle car une maille ne peut être reliée qu'à une seul autre maille.

c. Chaque point (maille) d'évacuation d'eau d'une parcelle constitue, avec la surface de la parcelle qui l'alimente, un exutoire. Les écoulements entre exutoires sont définis en s'appuyant sur le réseau de drainage.

La simulation des flux d'eau infiltrés et ruisselés par exutoire repose sur une démarche proche de celle du modèle STREAM (voir sous-section 2.1.1). Puis les flux sont agrégés par sommation sur les exutoires de parcelles empruntés au cours du transfert vers le réseau hydrographique. Cette sommation tient compte de possibles ré-infiltration de l'eau et des pesticides d'une parcelle amont à une parcelle aval.

2.1.2.2 Écoulements de subsurface

La structuration du profil du sol repose sur un découpage de celui ci en des colonnes ayant leur propre profondeur de toit de nappe. Le modèle s'appuie sur l'estimation journalière de la profondeur du toit de la nappe afin de simuler les écoulements de subsurface et sur surface saturée. Pour cela, l'indice topographique de Beven (équation 2.3) est utilisé, ainsi que le débit à l'exutoire du bassin versant.

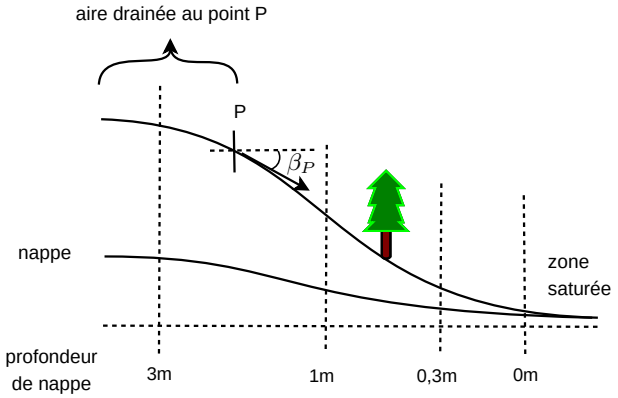


FIG. 2.6 – Au point P, qui peut être un point d'évacuation d'un exutoire de parcelle exu, l'indice de Beven est calculé à l'aide de la pente locale et de l'aire drainée. La profondeur de nappe est évaluée pour tous les jours de simulation et l'abattement des herbicides arrivant à la nappe est fonction de la colonne d'appartenance de exu.

2.1.3 Modélisation du transfert de pesticides

Au cours du temps, les matières actives à la surface se dégradent. La dégradation, en fonction du temps, est calculée à partir d'un paramètre de demi-vie de dissipation DT50 de la matière active du pesticide. Ce paramètre est le temps nécessaire pour réduire de moitié une quantité initiale donnée. Plus le DT50 d'une matière active est important, plus celle-ci met du temps à se dégrader et donc plus son utilisation comporte de risques pour le transfert.

Pour l'ensemble des écoulements modélisés, un coefficient de partage eau/sol est introduit pour déterminer les flux de pesticides couplés aux flux d'eau. Lors d'évènements pluvieux, une partie de la quantité d'herbicide présent sur la surface de l'*exutoire* est mise en solution avec l'eau ruisselante. La fonction d'échange permet d'évaluer cette quantité qui dépend d'un paramètre de rétention KOC de la matière active. Plus le KOC d'une matière active est faible, plus la quantité mise en solution est importante et donc plus le risque de transfert est grand.

Le risque de transfert, lié à l'utilisation d'une matière active donnée, peut finalement être estimé à partir de son DT50, de son KOC et des doses appliquées. Une classification des matières actives (risque fort, moyen ou faible[2]) est définie à l'aide de ces trois critères.

Les flux d'herbicides transférés par écoulement de subsurface à un exutoire donné sont abattus d'un facteur propre à la zone d'appartenance de l'exutoire : quatre zones sont en effet prédéfinies en fonction de la profondeur de la nappe (voir figure 2.6), la localisation de ces zones évoluent donc dans le temps. Par exemple, si l'exutoire, pour un jour de simulation donné, appartient à une zone proche du toit de la nappe ($<$0.30m) alors l'abattement de la quantité de pesticides lessivée est relativement faible (90% des quantités sont abattues) ; au contraire, si l'exutoire est en haut de bassin versant (nappe profonde) alors l'abattement est total (100%).

Alors que le transfert du ruissellement des pesticides est simulé au pas de temps journalier (c'est un processus rapide d'écoulement), les quantités de pesticides arrivant à la nappe par subsurface transfèrent vers le réseau hydrographique plus lentement. Un processus de "vidange" de la nappe est simulé, il dépend du débit d'eau journalier à l'exutoire du bassin versant. Le transfert de pesticides par subsurface vers le réseau hydrographique peut prendre plusieurs jours, il comprend un transfert couplé à la recharge de la nappe puis une vidange de celle-ci.

2.1.4 Modélisation des processus décisionnels

La modélisation des processus décisionnels doit permettre de simuler les caractéristiques des applications (produits, doses et dates) pour chacune des parcelles cultivées en maïs. La finalité du modèle décisionnel est d'introduire une variabilité spatiale et temporelle des applications en cohérence avec la réalité. Un *itinéraire technique* est, de manière générale, l'ensemble d'actes techniques du ressort de l'agriculteur qui décrivent son activité. Dans le cadre de cette thèse, nous nous intéressons uniquement aux actes liés au désherbage des parcelles cultivées en maïs.

Définition 2.7 (Itinéraire technique de désherbage - ITK).
Un itinéraire technique de désherbage (ITK) définit pour une parcelle maïs la stratégie d'application, celle-ci conditionnant les fenêtres temporelles d'application du pesticide. Une stratégie de type "prélevée puis post levée" consiste en deux applications, la première juste après le semis du maïs et la seconde au stade 5 feuilles du maïs. Une stratégie de

[2]Nous utilisons la classification des matières actives proposée par la CORPEP (Cellule d'Orientation Régionale pour la Protection des Eaux contre les Pesticides) http ://draf.bretagne.agriculture.gouv.fr/corpep/.

type "tout en post levée" consiste aussi en deux applications, la première au stade 3 feuilles du maïs et la seconde au stade 5-7 feuilles. L'ITK définit aussi les matières actives utilisées et donc la classe de risque de transfert associée (fort, moyen ou faible).

Les processus décisionnels sont modélisés dans le cadre de fenêtres temporelles d'action prédéfinies en fonction des pratiques agricoles types. Les actions[3] ne sont possibles qu'à l'intérieur des fenêtres temporelles prévues.

Les processus de décision sont simulés en fonction de contraintes formalisées. Les contraintes liées au temps de travail et à la disponibilité des machines vont étaler les actions dans le temps, sur la fenêtre temporelle prédéfinie (stratégie de l'ITK), et dans l'espace, sur la sole de maïs de chaque agriculteur. Les contraintes liées au climat vont différer l'action, ce report conduisant à concentrer ou étaler les actions sur l'ensemble du bassin versant. Les contraintes de milieu (parcelle humide de bas de versant par exemple) vont conduire à différer l'action sur certains secteurs du bassin versant. Les contraintes sont exprimées à l'aide de règles de décision. Les règles de décision, présentées sous la forme [*action A* si les *conditions B* sont respectées], sont adaptées à déterminer l'occurrence d'une action face à l'aléa climatique ou organisationnel (voir l'exemple 2.8).

Exemple 2.8 (règles de décision du modèle décisionnel).
Une application de pesticide peut être exécutée à un jour J sur une parcelle P si les conditions suivantes sont remplies :

- J est contenu dans la fenêtre temporelle prévue par l'ITK pour cette application ;
- la quantité de pluie au jour J est inférieure à un seuil prédéfini ;
- depuis le dernier jour de pluie, le temps écoulé est suffisant pour un réessuyage du sol assurant une condition de portance du sol et permettant aux engins agricoles de rentrer sur la parcelle P ;
- le temps nécessaire au traitement de la parcelle P (lié à la surface de P) ne nécessite pas un dépassement trop important du temps de travail journalier de l'exploitant ;
- un engin agricole est à disposition de l'exploitant pour l'application.

Le modèle SACADEAU s'appuie sur le couplage du modèle décisionnel et du modèle biophysique de transfert (figure 2.9).

Une simulation calcule les quantités d'herbicides transférées journalières sur la saison culturale, celle-ci commence en avril et s'étale sur quatre mois. Le fonctionnement général du modèle est donné dans l'algorithme 2.10.

2.2 Représentation des résultats de simulation

L'objectif ici est tout d'abord de faire un bilan des entrées et sorties du modèle. Les entrées sont catégorisées dans la sous-section 2.2.3. Des choix spécifiques de représentation sont donnés tels que le découpage du bassin versant en *arbres d'exutoires*

[3] ici, on ne considère que l'action de l'application de pesticides, mais, par exemple, le modèle requiert aussi de fixer la fenêtre temporelle de l'action du semis du maïs.

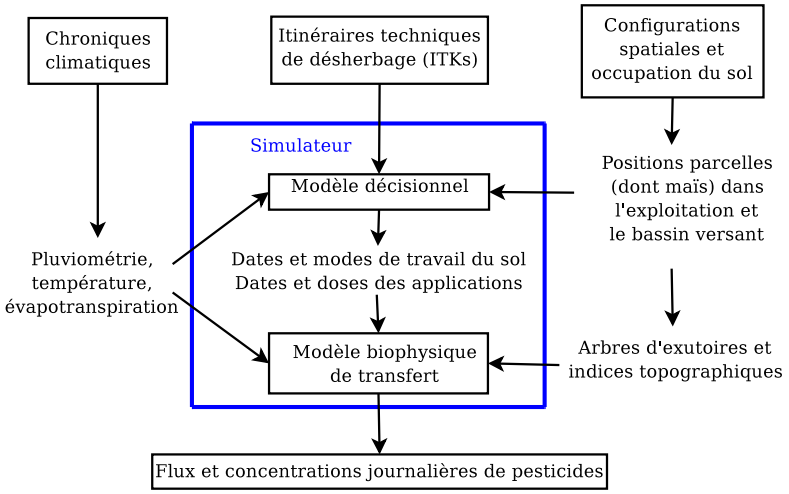


FIG. 2.9 – Le modèle SACADEAU est le couplage d'un modèle décisionnel simulant les pratiques agricoles et d'un modèle biophysique des transferts de l'eau et des herbicides.

Algorithme 2.10 : Fonctionnement général du modèle SACADEAU.

Entrées : - (voir sous-section 2.2.3)
Sorties : - quantités transférées et concentrations journalières pour chacune des matières actives

pour chaque *jour J de la saison culturale* **faire**
  pour chaque *parcelle P cultivée en maïs* **faire**
    si *une activité agricole A est en attente sur P et que les conditions pour A sont remplies* **alors**
      simuler A (désherbage, semis ...)
      (modèle décisionnel)
  pour chaque *exutoire de parcelle* **faire**
    simuler les transfert d'herbicides
    par ruissellement et subsurface
    (modèle de transfert)

(sous-section 2.2.2). Cette représentation est utilisée pour l'induction de règles dans le chapitre 3. Enfin, on se pose le problème de la caractérisation du transfert ; dans cette optique, on définit des variables de sorties du modèle (sous-section 2.2.3).

2.2.1 Entrées du modèle

Les différentes entrées du modèles sont regroupées, afin de les identifier par la suite, de la manière suivante :

– Les *données topologiques* rassemblent les données relatives à la connectivité des

parcelles, c'est-à-dire les relations d'écoulements entre exutoires, les données d'indice topographique et de pente de chaque exutoire, et les données relatives à la présence ou non d'un bord de champs ayant un rôle tampon ;
- Les *données d'occupation du sol* représentent l'affectation pour chaque parcelle d'une occupation du sol (maïs, céréale, prairie temporaire ...) ;
- Les *données d'exploitation* fournissent le regroupement en exploitations des parcelles cultivées en maïs ;
- Les *données décisionnelles* correspondent à l'affectation d'un itinéraire technique de désherbage (ITK) à chaque parcelle maïs ;
- les *données climatiques* sont les chroniques climatiques pour la saison culturale du maïs. Il s'agit en particulier de séries temporelles, à l'échelle de la journée ou de l'heure, des quantités de pluies et des températures.

2.2.2 Arbre d'exutoires

La modélisation mono-directionnelle du ruissellement implique qu'un exutoire de parcelle se déverse dans au plus un exutoire d'une autre parcelle ; elle permet la représentation du bassin versant à l'aide de structures d'arbres où les nœuds sont des exutoires et les arcs des relations entre eux.

Définition 2.11 (Relations "fils" et "en amont" entre exutoires).
L'exutoire de parcelle exu_B est un "fils" de l'exutoire exu_A (ou exu_A est le père de exu_B) si exu_B se déverse dans exu_A. Un exutoire exu_C est dit en amont[4] *de l'exutoire exu_A (ou exu_A est en aval de exu_C) s'il existe un chemin entre exu_C et exu_A, autrement dit, s'il existe un ensemble d'exutoires $exu_1, .., exu_n$ tels que exu_C est le fils de exu_n, exu_{i+1} est le fils de exu_i (pour tout i) et exu_1 est le fils de exu_A.*

Définition 2.12 (Arbre d'exutoires et sa racine).
Un exutoire exu_R se déversant dans le réseau hydrographique ou ne se déversant dans aucun exutoire est une racine d'arbre d'exutoires. L'arbre d'exutoires de racine exu_R est le sous-bassin versant composé de exu_R et des exutoires en amont de exu_R.

Nous proposons un découpage du bassin versant en "sous-bassins versants" qui sont des arbres d'exutoires. Les sous-bassins alimentent tous le réseau hydrographique par subsurface. Seuls ceux ayant un lien entre leur racine et le réseau hydrographique alimentent ce réseau par ruissellement (voir figure 2.13). On remarque que la description des exutoires d'un arbre peut s'appuyer sur des variables internes du modèle. Notamment, les exutoires de l'arbre représenté pour l'exemple dans la figure 2.14 sont décrits, lorsqu'ils font partie d'une parcelle cultivée en maïs, par la quantité totale de matière active appliquée sur la surface de l'exutoire. Cette variable n'est pas une entrée du modèle, elle est établie par le modèle décisionnel. Elle représente néanmoins un facteur d'intérêt pour le problème du transfert d'herbicides.

Pour un exutoire donné, tout flux le traversant provient soit d'exutoires amont soit de la surface de parcelle qui l'alimente. Ainsi, en s'intéressant à un arbre d'exutoire, on

[4] La relation "en amont" est en fait la clôture transitive de la relation "fils".

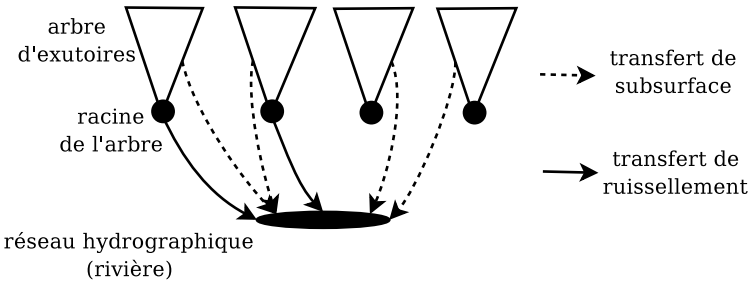


FIG. 2.13 – Représentation d'un bassin versant par un ensemble d'arbres d'exutoires. Tous les arbres alimentent le réseau hydrographique par subsurface, certains l'alimentent aussi par ruissellement.

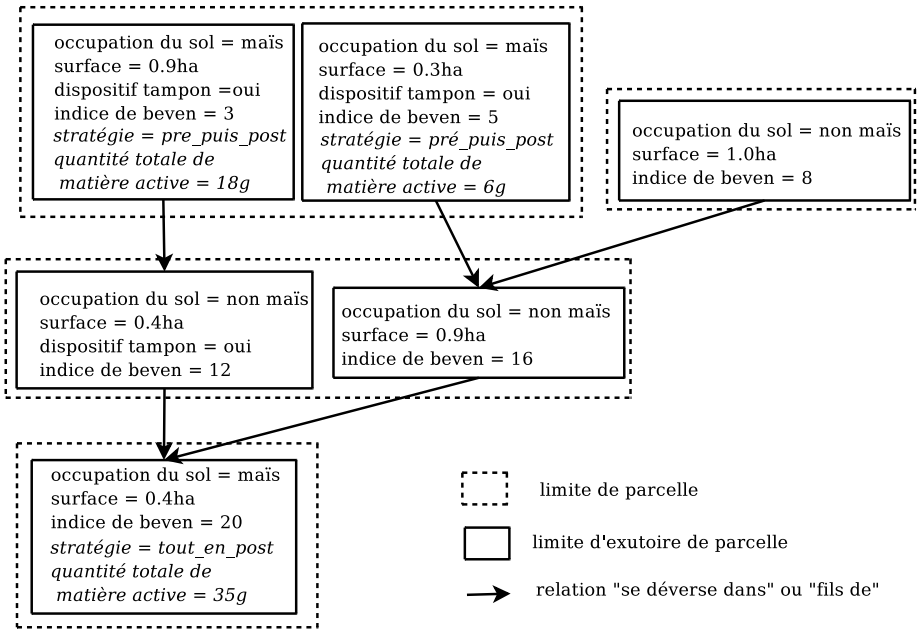


FIG. 2.14 – Un exemple d'arbre d'exutoires où chaque exutoire est décrit à l'aide de données d'occupation du sol, topologiques et décisionnelles (en italique). Cette description est formalisée par un ensemble d'attributs dans le chapitre 3.

peut estimer la contribution de ce sous bassin versant en termes de transfert d'herbicides, et ceci indépendemment des autres arbres : il n'y a pas de transfert entre deux arbres d'exutoires.

2.2.3 Variables de sortie

Les sorties de simulation sont les concentrations journalières, pour chacune des substances actives et pour chaque jour de la saison culturale (de avril à août), à l'exutoire du bassin versant. Dans l'optique d'analyser les résultats de simulation, nous définissons des variables afin d'agréger ces sorties. Deux points de vues sont ici adoptés pour évaluer le transfert de pesticides. Le premier s'appuie sur le cumul, sur la saison culturale, des concentrations d'herbicides à l'exutoire du bassin versant ; le deuxième repose sur la définition d'une variable de taux de transfert dans le but d'estimer l'intensité de l'effet tampon global (voir ci-dessous) pour un arbre d'exutoires puis un bassin versant.

La concentration de substances actives dans l'eau à l'exutoire du bassin versant est un critère souvent utilisé pour estimer la qualité de l'eau. L'un des objectifs du programme Bretagne Eau Pure, par exemple, est de maintenir une concentration journalière inférieure à $0,5\ \mu g.l^{-1}$ pour le cumul des concentrations des substances actives présentes. On s'intéresse ici au cumul, pour toute substance active et tous les jours simulés, des concentrations journalières à l'exutoire du bassin versant. Ces cumuls de concentrations nous permettent de définir une classification des simulations effectuées (tableau 2.15).

Cumul des concentrations (en $\mu g.l^{-1}$)	< 50	≥ 50 < 200	≥ 200 < 500	≥ 500 < 1000	> 1000
Classe de concentration	$conc0$	$conc1$	$conc2$	$conc3$	$conc4$

TAB. 2.15 – Définition des classes de concentrations en fonction du cumul, pour les substances actives utilisées et sur la saison culturale du maïs, des concentrations à l'exutoire du bassin versant.

La variable $taux\_transfert$ a pour objectif quant à elle d'estimer l'intensité de l'effet tampon. Le concept d'effet tampon (voir par exemple (Viaud *et al.*, 2004)) exprime la résistance du paysage au transfert d'eau et/ou de charges polluantes. L'intensité de l'effet tampon se mesure par la relation entre flux d'entrées et de sorties. Un paysage présente un effet tampon si la charge polluante baisse de l'entrée à la sortie. Les phénomènes impliqués sont connus et susceptibles de se développer dans des structures très diversifiées (marais, haies, prairies ...).

Cette variable peut être calculée pour un arbre d'exutoire ; en effet, on a vu que ceux-ci sont indépendants entre eux vis-à-vis du transfert de pesticides (sous-section 2.2.2). Quelques notations sont nécessaires : soit un exutoire e, une substance active

$m \in M$ et une chronique climatique[5] c donnés, les valeurs suivantes sont des sorties de simulation :

- $apport_{c,m,e}$: la quantité de matière active m appliquée sur l'exutoire e ;
- $sub_{c,m,e}$: la quantité de matière active m qui est transférée au réseau hydrographique par subsurface au niveau de e ;
- $ruiss_{c,m,e}$: la quantité de matière active m qui ruisselle de e ;

Pour un arbre d'exutoires a, on note $exus(a)$ l'ensemble de ses exutoires et $rac(a) \in exus(a)$ son exutoire racine. Soit c une chronique climatique (voir sous-section), nous définissons le taux de transfert à l'échelle d'un arbre d'exutoires a en nous appuyant sur les quantités d'herbicides apportées et transférées :

$$apport(c,a) = \sum_{m \in M, e \in exus(a)} apport_{c,m,e}$$

$$transfert(c,a) = \sum_{m \in M} \left(ruiss_{c,m,rac(a)} + \sum_{e \in exus(a)} sub_{c,m,e} \right)$$

$$taux\_transfert(c,a) = \frac{transfert(c,a)}{apport(c,a)} \tag{2.16}$$

Nous généralisons cette variable pour estimer l'effet tampon sur l'ensemble du bassin versant. Soit une chronique climatique c et un bassin versant bv, on note $arbs(bv)$ l'ensemble des arbres d'exutoires constituant la bassin versant bv, le taux de transfert est défini par :

$$taux\_transfert(c,bv) = \frac{\sum_{a \in arbs(bv)} transfert(c,a)}{\sum_{a \in arbs(bv)} apport(c,a)} \tag{2.17}$$

Deux variables sont utilisées pour caractériser le transfert. La classe de concentration cumulée est une variable qualitative pour caractériser le transfert à l'échelle du bassin versant. Le taux de transfert est une variable quantitative qui est calculée à l'échelle de l'arbre d'exutoires et celle du bassin versant.

2.3 Exploration du modèle

Le modèle SACADEAU peut être utilisé dans le but d'étudier l'impact de certains facteurs sur les résultats de simulation. Les variables de sortie utilisées pour cette étude sont le taux de transfert (équation 2.17) et la concentration cumulée en pesticides à l'exutoire du bassin versant (tableau 2.15).

[5] Une chronique climatique décrit les conditions météorologiques journalières sur l'ensemble de la saison culturale du maïs. Il s'agit donc d'une séquence de données météorologiques concernant la pluviométrie, la température, etc..

Les simulations sont effectuées dans le cadre de *scénarios*, chacun ayant pour objectif de faire ressortir l'impact d'un facteur particulier sur les variables de sortie. Un scénario est vu ici comme un ensemble de contraintes sur les entrées du modèle. Un premier scénario, dit de référence, est défini (sous-section 2.3.2) et s'appuie sur les données disponibles pour la simulation (sous-section 2.3.1). À partir de ce scénario, trois autres scénarios sont définis et concernent l'impact de la surface totale cultivée en maïs, celui de l'utilisation de matière active à risque de transfert fort, et celui de la connectivité des exutoires de parcelles cultivées en maïs (sous-section 2.3.3).

2.3.1 Données disponibles pour la simulation

Le bassin versant du Frémeur (Morbihan) a été choisi comme site d'application car c'est un bassin versant où des actions de remédiation sont en cours ; il est de petite taille, ce qui facilite l'acquisition des données. Il fait partie du programme Bretagne Eau Pure et nous disposons sur ce bassin de quantités importantes de données sur l'utilisation de produits phytosanitaires tels que les herbicides. Voici quelques données pour résumer la situation du bassin :

- il a une superficie totale de 14 km^2 (1413 ha) ;
- sa surface agricole utile (SAU) est de 78%, ce qui représente une forte pression agricole ;
- il est constitué de 1419 parcelles qui peuvent être divisées en 5312 exutoires de parcelles, il peut être représenté par 692 arbres d'exutoires ;

Des enquêtes, entre autres réalisées par Tortrat (Tortrat, 2005), nous permettent de disposer de l'ensemble des données relatives à l'occupation du sol et au désherbage pour l'année culturale 2000. Pour résumer :

- 30% de la SAU est cultivé en maïs, 30% en autres céréales, 25 à 30% en prairies et 10% en légumes ;
- 302 hectares sont cultivés en maïs (donnée $SMais$) ;
- 36 exploitations cultivent du maïs (donnée $NbExp$) ;
- 416 parcelles sont identifiées comme *potentiellement cultivables en maïs* (donnée PCM) ; ce sont les parcelles qui sont ou pourront être cultivées en maïs, par le fait des rotations culturales ; cela concerne les parcelles dont l'occupation (dans l'année culturale 2000) est soit en maïs, soit en prairie temporaire, soit en céréale ;
- 50 ITKs sont identifiés dont 30 sont de type "prélevée puis post levée" et 20 de type "tout en post levée", les ITKs utilisent en tout une quinzaine de matières actives ;

Des stations météorologiques proches du bassin versant nous ont fourni 28 chroniques annuelles climatiques, étalées sur une période de dix années, dont 9 ont été receuillies entre 1994 et 2002 à la station de Naizin (la plus proche du bassin versant du Frémeur).

2.3.2 Scénario de référence

Un scénario est exprimé par un ensemble de contraintes sur les entrées du modèle. Il est en effet infaisable d'énumérer puis de simuler toutes les combinaisons d'entrées

possibles. Par exemple, de nombreuses entrées sont des variables réelles, donc de domaine infini. Ici, un premier scénario est défini, il repose d'une part sur les données recueillies pour la saison culturale de l'année 2000 afin de produire, dans un premier temps, des simulations assez proches du terrain (voir section 2.3.1). Les contraintes de ce scénario sont les suivantes :

- **contrainte 1** : les chroniques climatiques pour la simulation sont des chroniques parmi celles recueillies, sur 9 années, à la station météorologique de Naizin ;
- **contrainte 2** : les données topologiques et d'écoulement sont celles du bassin versant du Frémeur ;
- **contrainte 3** : la surface totale de maïs sur le bassin égale celle des données recueillies que l'on note $SMais$ à un pourcentage ERR près ;
- **contrainte 4** : les parcelles maïs sont nécessairement des parcelles identifiées comme potentiellement cultivables en maïs lors de la saison culturale 2000 ;
- **contrainte 5** : une parcelle s'écoulant directement dans le réseau hydrographique ne peut être cultivée en maïs que si celle-ci dispose d'une bande enherbée, ceci dans le respect de la réglementation ;
- **contrainte 6** : le nombre d'exploitations est une constante $NbExp$ et correspond au nombre d'exploitations cultivant du maïs sur le bassin versant du Frémeur au cours de la saison 2000 ;
- **contrainte 7** : une exploitation possède un ensemble de parcelles maïs proches les unes des autres ;
- **contrainte 8** : une grosse exploitation (qui cultive plus de 10 hectares de maïs) met en œuvre au plus deux ITKs sur l'ensemble de ses parcelles cultivées en maïs : un ITK de type "prélevée puis post levée", un autre de type "tout en post levée". Une petite exploitation, par contre, ne met en œuvre qu'un seul ITK. La base d'ITKs disponible est celle recueillie au cours de la saison culturale 2000 sur le bassin versant du Frémeur.

En pratique, une combinaison d'entrées du simulateur est générée dans un processus de 4 étapes : affecter une occupation du sol à chaque parcelle du bassin versant (données d'occupation du sol), créer un ensemble d'exploitations se répartissant les parcelles cultivées en maïs (données d'exploitations), affecter à chaque parcelle maïs un ITK (données décisionnelles) et enfin choisir une chronique climatique (données climatiques).

Pour déterminer la localisation des parcelles maïs, un système de contraintes est mis en place. Notons PCM, l'ensemble de taille n des parcelles potentiellement cultivables en maïs et P_i une variable dans $\{0, 1\}$ égale à 1 si la $i$ème parcelle de PCM est une parcelle maïs pour la combinaison d'entrées à générer (et 0 sinon). Afin de respecter les contraintes 3, 4 et 5, on se base sur une solution du système de contraintes suivant[6] :

$$\left[\begin{array}{l} SMais * (1 - ERR) \leq P_1 * s_1 + \ldots + P_n * s_n \leq SMais * (1 + ERR) \\ \forall i \in NM, P_i = 0 \end{array} \right. \quad (2.18)$$

Où $P_1, \ldots, P_n$ sont les variables libres, et NM est l'ensemble des parcelles potentiellement cultivable en maïs, dont un exutoire, ne possédant pas de bande enherbée, se jette

[6] Le système de programmation par contraintes utilisé est le système Choco (Laburthe, 2000).

directement dans le réseau hydrographique. Dans le but d'introduire de l'aléa dans la génération des combinaisons d'entrée, l'ordre de l'évaluation des variables P_i pour la résolution du système est fixé aléatoirement.

Le groupement des parcelles maïs en exploitations est réalisée à l'aide d'une technique de regroupement (traduit de l'anglais *clustering*) où les données, représentées par leur coordonnées géographiques, représentent les parcelles maïs identifiées lors de l'étape précédente. Au vu de la contrainte 7, cette technique permet en effet de s'assurer que les parcelles maïs d'une même exploitation sont proches les unes des autres. Le choix de l'algorithme $k-means$ (MacQueen, 1967) est motivé d'une part, par la possibilité de fixer le nombre de cluster à $NbExp$ (contrainte 6) et d'autre part parce qu'il introduit naturellement de l'aléa dans les ensembles d'exploitations générées[7].

2.3.3 Étude de l'impact de certains facteurs

L'objectif ici est de s'intéresser à l'influence de certains facteurs sur le taux de transfert et la concentration cumulée en pesticides à l'exutoire d'un bassin versant. Pour cette étude, des modifications au scénario de référence sont apportées : par exemple, pour analyser les effets de la surface cultivée en maïs, la contrainte 3 est relâchée. Des contraintes cependant, sont difficiles à relâcher. Ainsi, la contrainte 2 ne peut être relâchée uniquement que si l'on possède les données topologiques pour un autre bassin versant ou un générateur de ces données ; dans les deux cas, cela demande de fournir un travail considéré trop important pour le cadre de cette thèse.

Trois facteurs sont étudiés de manière indépendante. La génération des combinaisons d'entrées du modèle, pour chacun des scénarios, intègre les modifications nécessaires.

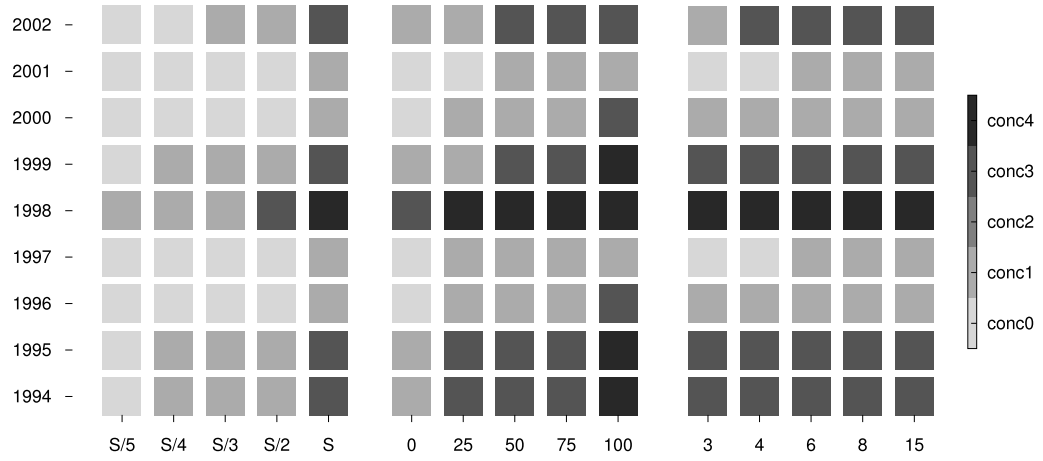
Sur la surface totale cultivée en maïs (**scenario1**) : le facteur d'intérêt de ce scénario est la surface totale cultivée en maïs. La contrainte 3 est modifiée, les valeurs fixées pour cette variable sont $\{SMais/5, SMais/4, SMais/3, SMais/2, SMais\}$.

Sur l'utilisation de substances actives à risque de transfert fort (**scenario2**) : le facteur d'intérêt pour ce scénario est le pourcentage de parcelles cultivées en maïs pour lesquelles l'ITK déployé est basé sur l'utilisation de substances actives à risque de transfert fort. La contrainte 8 est alors modifiée et nous faisons varier cette variable dans l'ensemble $\{0\%, 25\%, 50\%, 75\%, 100\%\}$. Une valeur de 0% signifie qu'aucune substance active au risque de transfert fort n'est présente sur l'ensemble du bassin versant.

Sur le placement des parcelles cultivées et la connectivité des exutoires (**scenario3**) : le facteur d'intérêt pour ce scénario est le nombre maximal de fils d'un exutoire de parcelle cultivée en maïs. L'objectif de ce scénario est de montrer l'impact de la localisation des parcelles en maïs sur le transfert, en particulier si celles-ci se situent à l'intersection de plusieurs chemins de flux d'eau. En pratique, cela revient à ajouter de nouvelles contraintes au système défini dans l'équation 2.18. Cette variable peut prendre les valeurs $\{3, 4, 6, 8, 15\}$. Si l'on fixe cette variable à 3, toute parcelle possédant un exutoire qui a plus de 3 fils est contrainte à ne pas être cultivée en maïs. Fixer la

[7] L'implémentation de $k-means$ utilisée est celle de WEKA (Witten & Frank, 2005).

variable d'intérêt à 3, 4, 6, 8 ou 15 contraint respectivement 25%, 14%, 6%, 2% ou 0% des parcelles potentiellement cultivables à ne pas être cultivées en maïs.



(a) **scenario1** : surface totale cultivée en maïs ($S = SMais$)

(b) **scenario2** : % d'utilisation de substances actives à risque fort

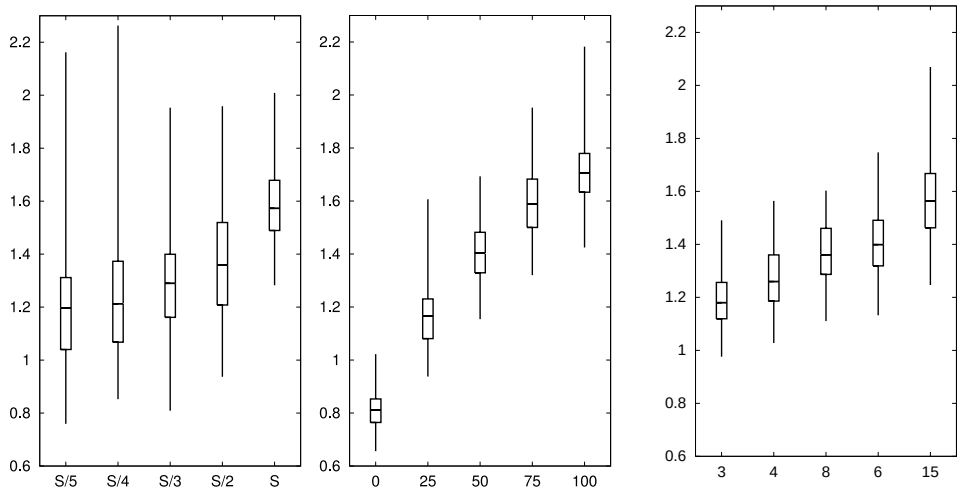
(c) **scenario3** : nombre de fils maximal pour un exutoire de parcelle maïs

FIG. 2.19 – Les classes de concentrations majoritaires (tableau 2.15) sont illustrées en fonction de la valeur des paramètres de scénario et de la chronique climatique utilisées pour la simulation. Un carré représente la classe majoritaire pour 80 simulations.

Pour chacune des valeurs de paramètre d'un scénario, 80 combinaisons d'entrées partielles (comprenant les données décisionnelles, d'exploitation et d'occupation du sol) ont été générées. Pour chacune d'elles et chacune des chroniques climatiques (contrainte 1), une simulation a été effectuée.

Les résultats des simulations, reposant sur la classe de concentration présentée dans le tableau 2.15, sont illustrés dans la figure 2.19. Les données climatiques jouent un rôle primordial sur la concentration cumulée à l'exutoire du bassin versant : par exemple, la chronique climatique de 1998 conduit à une concentration cumulée importante quelle que soit la valeur de paramètre fixée pour les deux derniers scénarios. L'influence de la surface totale de maïs cultivée est la plus marquée pour les trois scénarios considérés. Ceci est tout à fait normal, en effet, les quantités de pesticides apportées, et donc celles transférées, diminuent nécessairement avec la surface cultivée en maïs. De même, le risque de transfert d'une substance active dépend des doses apportées. Les résultats montrent également un impact relativement important de l'utilisation de matières actives à risque fort de transfert, celle-ci est en général appliquée à de plus fortes doses. En revanche, l'impact du placement des parcelles cultivées est moins clair.

Nous proposons également de visualiser le taux de transfert (équation 2.17), moyenné sur les chroniques climatiques disponibles, en fonction des valeurs des paramètres de



(a) **scenario1** : surface totale cultivée en maïs ($S = SMais$)

(b) **scenario2** : % d'utilisation de substances actives à risque fort

(c) **scenario3** : nombre de fils maximal pour un exutoire de parcelle maïs

FIG. 2.20 – Taux de transfert (équation 2.17) moyennés sur les chroniques climatiques en fonction de la valeur des paramètres de scénario. Pour une valeur de paramètre de scénario, cette illustration permet d'identifier sur l'axe des y, la valeur minimale, le seuil pour lequel 25% des points sont en dessous, la valeur moyenne, le seuil pour lequel 75% des points sont en dessous et la valeur maximale.

scénario (voir figure 2.20). On remarque que la surface de maïs influe de manière moins importante. Par contre l'effet du placement des parcelles maïs et de l'utilisation de substances actives à risque fort ressortent plus clairement. On remarque également que dans le cadre du premier scénario, le taux de transfert varie plus fortement que dans pour les deux autres scénarios. Il existe donc des facteurs déterminants pour le taux de transfert autres que la surface cultivée.

Nous tirons deux conclusions de cette étude. Tout d'abord, le rôle important des données climatiques nécessite une attention particulière. Par exemple il peut être intéressant de définir une typologie des climats pour le problème de transfert de pesticides (chapitre 3). D'autre part, il existe un impact de la connectivité des parcelles et du type de substance active utilisée sur un taux de transfert moyenné sur les chroniques climatiques.

2.4 Conclusion

Dans le cadre du projet SACADEAU, un modèle de transfert de pesticides à l'échelle d'un bassin versant a été développé. Ce modèle a privilégié les variables observables fa-

cilement sur le terrain (états de surface du sol, exutoires de parcelle ...) pour permettre l'information et l'appropriation du modèle. La finalité de coupler un modèle décisionnel au modèle de transfert est tout d'abord de représenter les activités liées au désherbage du maïs et aussi, d'introduire une variabilité spatiale et temporelle des applications en cohérence avec la réalité.

Des représentations des données de simulation, dont les entrées et sorties de modèle, sont spécifiées. Ainsi, des variables de transfert sont définies afin d'agréger les sorties de simulations ; en effet, les sorties sont nombreuses, elles sont en particulier les quantités transférées et concentrations d'herbicides journalières sur toute la saison culturale du maïs. Un mode d'analyse du bassin versant reposant sur la structure d'arbres d'exutoires, sous-bassins versants indépendants vis-à-vis du transfert, a été développé.

Enfin une étude, réalisée à partir des résultats de simulations définies dans le cadre de scénarios, montre l'impact des données climatiques sur la concentration cumulée à l'exutoire du bassin versant. En considérant un taux de transfert, l'impact des données décisionnelles, d'occupation du sol et de topologies est mis en avant. La localisation des parcelles maïs à des endroits stratégiques et l'utilisation de matières actives à risque faible peuvent être des moyens d'agir sur cette variable de sortie.

Chapitre 3

Induction de règles à partir des données issues du modèle SACADEAU.

L'objectif de l'apprentissage de règles dans ce travail de thèse est d'expliquer les résultats de transfert de pesticides issus de simulations réalisées avec le modèle SACADEAU, en fonction des facteurs déterminants pour la prise de décision. Les règles alors obtenues doivent permettre de dégager les variables explicatives du transfert sur lesquelles des actions peuvent être conduites.

Les problèmes que nous nous posons au préalable sont la spécification des étiquettes de sortie du modèle (quelle est la variable de sortie pertinente ?) et celui de la représentation des entrées de simulation (comment faut-il décrire les données de simulation ?). Autrement dit, pour l'induction de règles de classification, il est nécessaire de définir les classes d'apprentissage, de définir la description des exemples, c'est-à-dire le langage des exemples $\mathcal{L}_E$ et enfin de formaliser les conditions présentes dans les règles, c'est-à-dire le langage des hypothèses $\mathcal{L}_H$ (voir chapitre 1). De manière générale, lors d'un processus d'acquisition automatique de connaissances, la formulation du problème d'apprentissage est une étape importante.

Un premier problème d'apprentissage a été étudié dans le cadre du projet SACADEAU (section 3.2). Les classes d'apprentissage reposent sur la concentration de pesticides à l'exutoire du bassin versant. Les résultats de simulation sont issus d'un modèle SACADEAU simplifié. Dans ce travail, nous nous sommes intéressés à l'échelle du bassin versant et les données climatiques y ont été particulièrement étudiées.

Le problème d'apprentissage plus largement approfondi dans ce chapitre est la caractérisation du taux de transfert des arbres d'exutoires (section 3.3). Nous faisons l'hypothèse qu'il existe un impact des données décisionnelles, de topologie et d'occupation du sol sur le taux de transfert, ce que suggère également l'étude du chapitre 2. Ces facteurs sont déterminants pour l'aide à la prise de décision. La connectivité des parcelles (via leurs exutoires) est difficile à étudier à l'échelle du bassin versant, c'est pourquoi nous considérons les arbres d'exutoires, sous-bassins versants indépendants vis-à-vis du

transfert (voir définition 2.12). Dans la section 3.3, une base d'apprentissage est constituée d'exemples représentant des arbres d'exutoires et le terme de caractérisation est précisé pour nos apprentissages.

Pour l'induction, une première approche consiste à conserver la structure d'arbre dans la représentation de nos exemples d'apprentissages (section 3.4). La logique des prédicats permet d'exprimer les relations entre exutoires et donc de manipuler des structures d'arbres, elle est utilisée afin de spécifier les langages $\mathcal{L}_E$ et $\mathcal{L}_H$. La deuxième approche consiste à définir des attributs globaux, dits agrégats, afin de décrire les données, exemples et règles, dans une logique attribut-valeur. La dernière approche consiste à combiner les deux premières (section 3.6).

En premier lieu, dans la section 3.1, nous présentons les problématiques de différentes applications pour l'apprentissage de règles à partir de données issues de simulations.

3.1 Induction de règles à partir de données de simulation

L'apprentissage d'une théorie à partir des résultats de simulation peut être vue comme la construction d'un méta-modèle du simulateur (Pierreval, 1992). Deux techniques pour la méta-modélisation sont distinguées : les approches statistiques et les approches basées sur l'apprentissage automatique (Merkuryeva, 2004). Les approches statistiques, par exemple (Kleijnen, 1979; Friedman, 1996), sont basées sur des analyses statistiques des sorties de simulation telles que la régression linéaire (voir exemple 3.1). Les approches basées sur l'apprentissage automatique utilisent des techniques telles que les réseaux de neurones ou l'apprentissage de règles et peuvent s'appuyer sur une représentation en logique floue ; elles sont moins développées dans la littérature.

Exemple 3.1 (la régression linéaire pour la création de méta-modèles).
Étant donné un modèle de simulation à n entrées $x_1, \ldots, x_n$, le résultat de la simulation du modèle avec $(x_1, \ldots, x_n)$ comme vecteur d'entrées est noté y. L'équation linéaire $y = \beta_1 * x_1 + \ldots + \beta_n * x_n + e$ peut être vu comme un méta modèle statistique du modèle d'origine. Il s'agit alors de construire le vecteur $(\beta_1, \ldots, \beta_n)$ tel que, pour un ensemble de simulations le plus complet possible, l'erreur e soit minimale.

Nous nous intéressons aux approches basées sur l'apprentissage de règles de classification (voir chapitre 1), c'est à dire des règles du type "si condition alors classe". De manière générale (Zhang *et al.*, 2002; Mladenic *et al.*, 1994; Huber & Berthold, 1997; Alexander & Kelly, 2006), un premier intérêt des méta-modèles à base de règles est la traduction dans un langage facilement interprétable des résultats de simulation afin de mieux les comprendre. Les conditions des règles (respectivement les classes d'apprentissage) dépendent des entrées du simulateur (respectivement de ses sorties). En particulier, dans cette section, nous nous intéressons aux deux points suivants :

- les objectifs d'un apprentissage de méta-modèle à base de règles ;
- la constitution d'un ensemble d'apprentissage à partir de données de simulations.

3.1.1 Objectifs

Dans le cadre de la construction d'un méta-modèle à base de règles, l'objectif pour l'apprentissage de règles peut différer de celui de la construction d'un classifieur : il ne s'agit plus de construire des classifieurs qui assignent la bonne classe d'apprentissage aux exemples non étiquetés ; en effet, si les temps de simulation ne constituent pas un obstacle, cette classification peut être effectuée à l'aide du simulateur (Mladenic *et al.*, 1994). Zhang et al. (Zhang *et al.*, 2002) suggèrent de se focaliser sur des règles de *caractérisation* en logique attribut-valeur. De leur point de vue, les règles de caractérisation s'écrivent sous une forme similaire aux règles de classification (section 1.1). Alors que les règles de classification sont utiles dans la classification d'exemples non étiquetés, les règles de caractérisation sont destinées à la compréhension par l'utilisateur de ce qui caractérise une classe d'apprentissage. En particulier, les règles redondantes (c'est à dire qui couvrent des exemples déjà couverts par d'autres règles) peuvent être inutiles pour la classification et utiles pour la caractérisation[1].

L'efficacité peut être aussi un intérêt de l'utilisation de méta-modèles à base de règles. Les systèmes à base de connaissances (KBS pour l'acronyme de l'anglais *knowledge based systems*) sont constitués de connaissances a priori dans un domaine et sont utilisés pour l'aide à la prise de décision de l'utilisateur. Afin de prendre en compte le caractère dynamique des systèmes étudiés dans les KBS, O'Keefe (O'Keefe, 1986) suggère d'y intégrer des modèles de simulation. Mais la simulation, dans certaines applications, est rendue difficile pour des raisons pratiques car elle nécessite un temps de calcul et un espace mémoire trop important (Merkuryeva, 2004; Pierreval, 1992). Pierreval (Pierreval, 1992) suggère alors au préalable, d'induire un ensemble de règles entre les entrées et sorties du modèle et ensuite, utiliser la théorie pour prédire les sorties de simulation. Pierreval utilise donc la théorie induite pour remplacer la simulation trop coûteuse.

3.1.2 Constitution d'une base d'apprentissage

Afin de définir une base d'exemples à l'aide d'un simulateur, il faut générer des combinaisons d'entrées du simulateur, les simuler puis les décrire, ainsi que les résultats de leur simulation, dans le langage souhaité. Nous nous intéressons plus particulièrement aux choix pris pour l'affectation des valeurs d'entrées du simulateur, c'est-à-dire pour la sélection des combinaisons d'entrée. On considère ici que les classes d'apprentissage sont déterminées à partir des sorties de simulation, avec une discrétisation de ces sorties si celles-ci sont des variables quantitatives.

En apprentissage classique à partir d'exemples, la base d'apprentissage est issue d'observations réelles : en général, toutes ces observations sont utilisées pour l'apprentissage. Lorsque les exemples sont issus d'un simulateur, la base est en général potentiellement infinie car le nombre de combinaisons d'entrées du simulateur est lui même potentiellement infini.

[1] pour plus de précisions, se référer à la sous-section 3.3.2

Néanmoins, si les entrées du simulateur sont des variables de domaines finis suffisamment petits, une génération exhaustive de la base d'apprentissage en simulant toutes les combinaisons d'entrées peut être réalisée (Mladenic *et al.*, 1994). Dans le cas contraire, il est nécessaire de définir des scénarios propres à l'application afin de générer les entrées de simulations, par exemple (Zhang *et al.*, 2002). Huber et Berthold (Huber & Berthold, 1997) génèrent une base d'exemples d'apprentissage en se focalisant sur les variables d'entrées les plus pertinentes pour leur application, ils font varier aléatoirement les valeurs des ces variables, les autres variables conservant une valeur fixe pour toutes les simulations.

Parisay et Khoshnevis (Parisay & Khoshnevis, 1994) définissent le problème puis proposent une méthode d'élaboration d'une base d'apprentissage à partir d'un simulateur. De leur point de vue, la base d'apprentissage doit respecter trois contraintes :

- les exemples doivent être représentatifs des combinaisons d'entrées possibles du simulateur ;
- la distribution des exemples dans chacune des classes ne doit pas nuire à l'apprentissage (il faut éviter une base d'apprentissage déséquilibrée) ;
- les exemples doivent être uniques (issus de combinaisons d'entrées différentes) et suffisamment différenciés.

3.2 Un premier apprentissage à l'aide du modèle simplifié

Lors d'une première étape du projet, un modèle simplifié a été utilisé. Les fonctions de transfert de ruissellement sont celles détaillées dans le chapitre 2. Par contre, l'agrégation à l'échelle du bassin versant est simple et n'utilise pas la modélisation des flux à l'aide d'arbres d'exutoires, mais un abattement des quantités d'herbicides en fonction de la forme du bassin versant et de la distance des parcelles au réseau hydrographique. Dans ce modèle, les chemins de l'eau de parcelle à parcelle ne sont pas représentés.

Le modèle simplifié possède six entrées de domaines qualitatifs :

- le taux de matière organique dans le sol : celle-ci influe sur l'infiltrabilité du sol et l'absorption des herbicides. En entrée du modèle, deux taux ont été choisis : 50 $g.kg^{-1}$ et 20 $g.kg^{-1}$.
- la topologie générale du bassin : nous avons considéré deux types de bassins versant, les bassins versant *concaves* et les bassins versants *convexes*. Les pentes proches du cours d'eau pour un bassin versant de forme concave sont moins importantes que celles d'un bassin de forme convexe. À chaque type de pente est associé un abattement sur le transfert ;
- le type d'ITK : celui-ci peut prendre deux valeurs qui sont *prélevée puis post levée* ou *tout en post levée* (voir définition 2.7).
- le type de matière active utilisée : celui-ci peut prendre deux valeurs qui sont *atrazine* et *nouvelle*. L'atrazine est une molécule utilisée à forte dose avant 2001, de nouvelles molécules ont depuis été introduites pour la remplacer.
- le pourcentage de bordure de cours d'eau disposant d'une zone tampon, type bande enherbée : une zone tampon abat une partie des quantités d'herbicides

ruisselantes. Deux valeurs extrêmes ont été considérées 0% et 90%.
- le type de climat prend une valeur dans l'ensemble $\{1, 2, 3, 4, 5\}$. Un climat de type 1 est considéré comme sec et un type 5 comme très pluvieux (voir ci dessous).

Les entrées du modèle simplifié étant de domaines qualitatifs suffisamment restreints, la base d'apprentissage peut être constituée des résultats de simulation issus d'une génération exhaustive des combinaisons d'entrées (voir sous-section 3.1.2). En utilisant les 28 chroniques climatiques disponibles (voir sous-section 2.3.1), réparties dans les 5 types de climat définis, nous avons donc effectué 896 simulations, chacune représentant un exemple d'apprentissage. Chaque exemple a été étiqueté par une classe basée sur la concentration de pesticides à l'exutoire du bassin versant. Une analyse de la théorie induite (Cordier *et al.*, 2005a) montre un impact important des données climatiques et du taux de matière organique sur la classe de concentration.

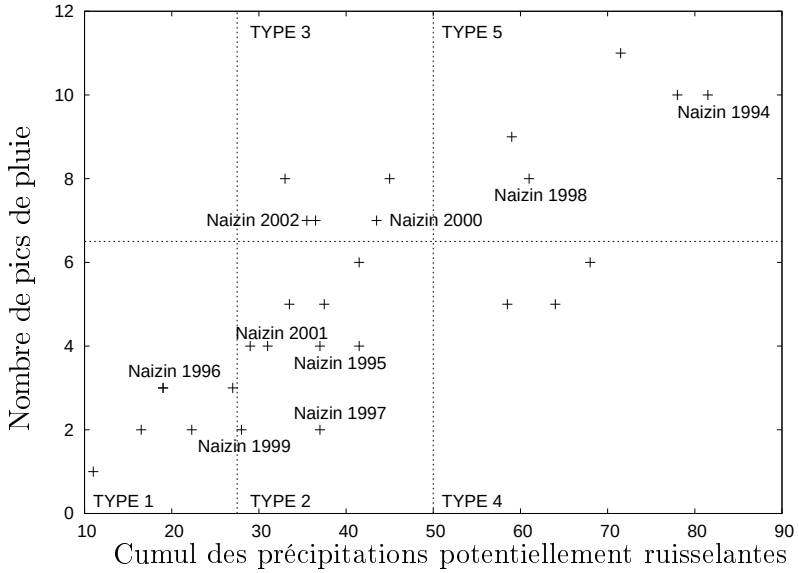


FIG. 3.2 – Les 28 chroniques climatiques (dont les 9 recueillies à la station de Naizin) et la typologie définie en fonction du cumul des précipitations potentiellement ruisselantes et du nombre de pics de pluie.

Au cours de ce travail, nous nous sommes particulièrement intéressés à la définition d'une typologie des chroniques climatiques en rapport avec le problème traité, c'est-à-dire le transfert de pesticides pour une saison culturale du maïs. Pour définir la typologie, nous avons utilisé une méthode de classification ascendante hiérarchique en comparant les chroniques climatiques disponibles entre elles ; et nous avons extrait 5 groupes de chroniques climatiques (de 1 à 5). La classification ascendante hiérarchique repose sur deux variables que nous avons jugé déterminantes : le nombre de jours où la pluie journalière a dépassé un seuil de 10mm, que nous appelons *pics de pluie* ; et le *cumul des précipitations potentiellement ruisselantes*, c'est-à-dire le surplus de la précipitation

par rapport à un seuil fixé à 2mm, cette quantité étant considérée comme le seuil au delà duquel il peut y avoir un ruissellement. Les chroniques climatiques ainsi typées sont illustrées dans la figure 3.2 en fonction de ces deux variables.

La typologie reflète, pour certaines chroniques, les résultats de simulation de l'étude du chapitre 2 (voir figure 2.19). Ainsi, les chroniques de la saison 1998 et 1994 de type 5 et donc pluvieuses impliquent des résultats de transfert important. Par contre, pour la chronique de la saison 1999, plutôt sèche, les résultats de transfert importants ne semblent pas correspondre.

D'autres facteurs, en effet n'ont pas été pris en compte dans cette typologie. Par exemple, on sait que la durée de la période entre une pluie et une application est un facteur important pour le transfert. Cette relation temporelle est difficilement exprimable à l'aide de variables comme celles que nous avons définies ici. Dans ce travail de thèse, nous n'étudions pas de telles relations. Cependant, dans le cadre du projet, des scénarios ont permis de s'intéresser à l'impact des relations temporelles. Ainsi, il a été montré qu'un retard des applications pouvait avoir lieu dans le cas où il existe de nombreuses précipitations autour des dates prévues, au sein des ITKs, pour les applications. Après ces pluies, l'ensemble des applications s'effectuent dans une période courte sur un sol saturé en eau, ce qui peut expliquer un transfert important.

3.3 Définition du problème d'apprentissage

L'objectif de l'apprentissage, pour le reste de cette thèse, est de se focaliser sur l'impact des facteurs suivants, déterminants pour l'aide à la décision :

- les itinéraires techniques de désherbage (données décisionnelles) ;
- la surface maïs et la localisation des parcelles cultivées en maïs (données d'occupation du sol) ;
- les configurations des chemins de l'eau, les conditions topologiques et les aménagements au niveau des parcelles (données topologiques).

Les données topologiques comprennent entre autres les données relatives à la connectivité entre exutoires, c'est à dire la structure d'arbre. Évaluer l'impact de telles relations est infaisable à l'échelle d'un bassin versant qui comprend plusieurs milliers d'exutoires. De plus, une étude à cette échelle nécessiterait de disposer des données d'un ensemble important de bassins versants afin de constituer une base d'exemples conséquente.

C'est pourquoi nous optons pour un découpage du bassin versant en sous-bassins versants : cela peut permettre de faire émerger des relations pertinentes de connectivité entre exutoires. La base d'exemples alors constituée est de taille suffisante. Afin d'évaluer le transfert à l'échelle d'un sous-bassin versant, il faut s'assurer que deux sous-bassins sont indépendants vis-à-vis du transfert : il ne doit pas y avoir d'échange d'eau ou d'herbicides entres deux sous-bassins.

Pour ces raisons, nous nous focalisons sur les arbres d'exutoires (section 2.2.2). La variable d'intérêt, dont le but est d'évaluer l'effet tampon d'un sous-bassin versant, c'est-à-dire la résistance du paysage au transfert, est le taux de transfert d'un arbre d'exutoires. Afin de limiter l'aléa lié au climat, on moyenne le taux de transfert d'un

arbre *arb* (équation 2.16) sur un ensemble de chroniques climatiques C (où $\|C\|$ est le nombre de chroniques climatiques) :

$$taux\_transfert(arb) = \sum_{c \in C} \left(\frac{taux\_transfert(c, arb)}{\|C\|} \right) \tag{3.3}$$

En premier lieu, une base d'exemples, constituée d'arbres d'exutoires, est générée à l'aide du modèle de simulations (sous-section 3.3.1). Dans la sous-section 3.3.2, nous formalisons la tâche d'apprentissage des règles de caractérisation du taux de transfert des arbres d'exutoires. En particulier on s'intéresse aux types d'algorithmes pour l'induction qui répondent le mieux à notre problème (voir sous-section 1.1.3).

3.3.1 La base d'arbre d'exutoires

Les simulations effectuées pour la constitution d'une base d'arbres d'exutoires sont déterminées à partir du scénario de référence défini en sous-section 2.3.2, la stratégie adoptée est donc celle de (Huber & Berthold, 1997) (voir sous-section 3.1.2). Nous avons, à l'aide de ce scénario, généré 20 combinaisons d'entrées partielles constituées des données topologiques, d'occupation du sol, d'exploitations et décisionnelles. Au total on extrait 3431 arbres d'exutoires sur lesquels il y a au moins un exutoire de parcelle maïs, et donc, pour lesquels il existe potentiellement un transfert d'herbicides depuis son application jusqu'au réseau hydrographique. En utilisant les 9 chroniques climatiques recueillies à la station météorologique de Naizin, le taux de transfert moyen (équation 3.3) a été calculé pour chacun des arbres d'exutoires extraits (figure 3.4).

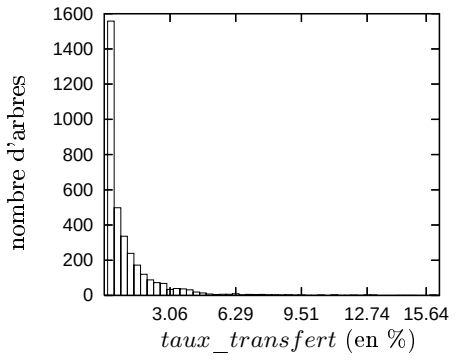


FIG. 3.4 – Histogramme du nombre d'arbres d'exutoires par tranche de taux de transfert moyen.

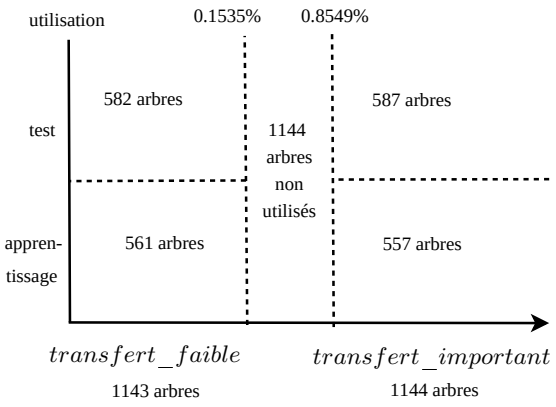


FIG. 3.5 – Répartition des arbres d'exutoires dans la base d'exemples selon leur taux de transfert moyen.

L'étiquetage des exemples, que nous souhaitons définir à l'aide du taux de transfert moyen, nécessite une discrétisation de cette variable. Nous avons fait les choix de ne pas déséquilibrer les classes (en terme de nombres d'exemples), et de ne considérer que les

taux de transfert extrêmes afin d'induire des règles pertinentes. On possède un nombre suffisant d'arbres d'exutoires pour en écarter certains de l'apprentissage. Pour cela, nous réalisons une partition de l'ensemble des arbres d'exutoires en 3 ensembles de taille égale en fonction de leur taux de transfert moyen (les deux valeurs frontières étant 0,1535% et 0,8549%), et nous définissons l'étiquette de classe d'un arbre d'exutoires de la manière suivante :

$$classe(arb) = \begin{cases} transfert\_faible & \text{si } taux\_transfert(arb) < 0,1535\% \\ transfert\_important & \text{si } taux\_transfert(arb) > 0,8549\% \end{cases} \quad (3.6)$$

Une base d'exemples est en général constituée des échantillons suivants [2] :

- *un échantillon d'apprentissage* : ce sont les exemples qui sont utilisés pour l'induction proprement dite des règles ;
- *un échantillon de validation* : ils permettent de calibrer le modèle, c'est à dire déterminer la valeurs des paramètres d'apprentissage ;
- *un échantillon test* : ceux-ci ne doivent en aucun cas être utilisés pour les étapes d'apprentissage ou de validation, ils permettent de comparer différentes théories induites, par exemple, en terme de taux de bonnes classifications. La classification est le procédé qui, à l'aide d'une théorie, assigne à un exemple sa classe.

Dans notre application, nous n'utilisons pas d'ensemble de validation, les paramètres d'apprentissage, peu nombreux, sont fixés de manière experte. Dans les deux groupes de classe, définis ci-dessus, nous tirons aléatoirement les exemples d'apprentissage et les exemples de tests (voir la base d'apprentissage en figure 3.5).

3.3.2 Caractérisation des classes d'apprentissage

De notre point de vue, s'il existe plusieurs règles qui couvrent un exemple d'apprentissage donné, les proposer toutes à l'utilisateur est un moyen de suggérer plusieurs explications possibles de la classe de l'exemple. Le problème peut alors être défini comme étant l'induction, pour chacune des classes, des règles correctes[3] qui couvrent un ensemble d'exemples non négligeable, ceci afin de généraliser ces exemples.

Michalski et al. (Michalski *et al.*, 1986b) différencient les théories *discriminantes* de celles *caractéristiques*. Une théorie discriminante doit respecter les conditions de complétude et correction, elle doit également consister en une description minimale permettant de discriminer des exemples de différentes classes. Elles ne sont pas adaptées à nos objectifs : notamment, si une théorie contient une règle dont les exemples couverts sont déjà couverts par d'autres règles, elle ne peut pas être discriminante car la description n'est pas minimale. Une théorie caractéristique doit respecter la condition de complétude ; dans ce cas, l'intérêt est porté sur une généralisation la plus spécifique des exemples d'une classe donnée, qui permet de distinguer cette classe des autres classes.

D'autres points de vue existent concernant les règles caractéristiques. Dans (McSherry & Roantree, 1999), ce sont des règles de classification $s_1 \wedge \ldots \wedge s_n \rightarrow c$ où chaque sélecteur s_i doit vérifier $p(c|s_i) > p(c)$ où $p(c|s_i)$, la probabilité de c sachant s_i, est

[2] voir (Cornuéjols & Miclet, 2002) pour plus de détails.

[3] voir définition 1.3

évaluée uniquement sur les exemples couverts par $s_1 \wedge \ldots \wedge s_{i-1}$. L'idée est, dans le processus de construction d'une règle, de n'ajouter un sélecteur à la règle seulement s'il est utile pour la classe c, et d'éviter l'ajout d'un sélecteur non déterminant. Dans (Turmeaux *et al.*, 2003), les règles caractéristiques sont des descriptions[4] qui doivent permettre de résumer de manière concise un ensemble de données ; leur apprentissage ne nécessite pas de contre exemples. Dans (Han *et al.*, 1993; Brijs *et al.*, 2000), le sens d'implication de la règle est inversé (on recherche des règles du type "classe implique complexe") et les apprentissages ne nécessitent également pas de contre exemples.

Finalement, nous adoptons le point de vue de Zhang et al. (Zhang *et al.*, 2002) présenté dans la section 3.1.1. Ils proposent d'induire des théories composées de règles de de caractérisation, qui sont en fait des règles de classification pouvant être redondantes, c'est-à-dire qu'une règle peut couvrir des exemples déjà couverts par d'autres règles. Nous clarifions le terme de caractérisation dans le cadre de nos apprentissages.

Définition 3.7 (Tâche de caractérisation des classes d'arbres d'exutoires). *La caractérisation d'une classe c d'arbres d'exutoires (voir équation 3.6) est l'induction des règles de classification de classe c correctes dont le support (nombre d'exemples couverts) est supérieur à un seuil fixé par l'utilisateur. Les exemples d'apprentissage sont des arbres d'exutoires (voir par exemple l'arbre d'exutoires en figure 2.14) étiquetés d'une des deux classes* $transfert\_important$ *ou* $transfert\_faible$.

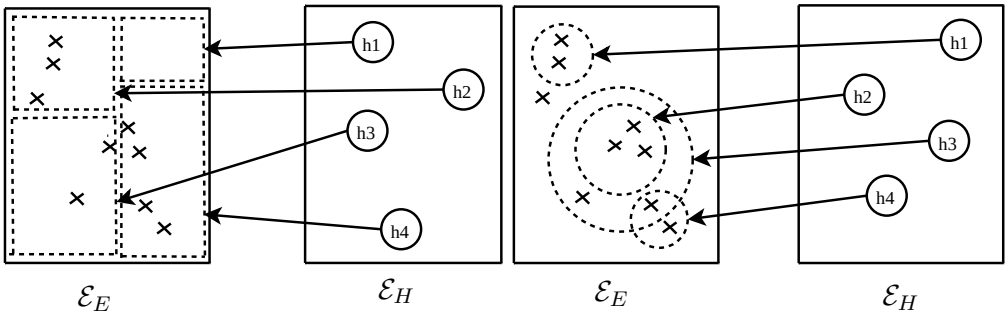
Le but de ce travail n'étant pas de développer de nouveaux algorithmes d'induction de règles, nous nous basons sur des algorithmes existants répondant au mieux au problème que l'on s'est fixé. Pour cela, les algorithmes de type "séparer pour régner" sont préférés à ceux de type "diviser pour régner" (voir sous-section 1.1.3) : ces derniers contraignent en effet un exemple à être couvert par exactement une règle (schéma 3.8).

En pratique, nous relâchons la contrainte de correction afin d'induire des règles non correctes mais qui couvrent peu d'exemples de classe opposée, elles peuvent en effet rester pertinentes. De plus, l'exploration de l'espace des hypothèses n'est pas complète, c'est pourquoi la théorie induite ne contient qu'un sous-ensemble des règles qui nous intéressent.

Enfin, différents travaux ont étudié la redondance des règles. Notamment, Zhang et Michalski (Zhang & Michalski, 1989) puis Torgo (Torgo, 1993) utilisent une mesure de règle, qu'ils appellent *utilité*, afin de générer des théories non redondantes. L'utilité d'une règle r est le ratio du nombre d'exemples que seule r couvre sur le nombre total d'exemples que r couvre ; en extrapolant, on peut considérer que la redondance d'une règle r est $1 - utilite(r)$. Dans une problématique de recherche de règles d'associations (voir l'exemple 1.2), de nombreux travaux, par exemple (Torgo, 1993; Brijs *et al.*, 2000; Jaroszewicz & Simovici, 2002), ont pour objectif, à l'opposé du notre, d'induire des théories non redondantes.

L'interprétation d'une théorie redondante peut être difficile du fait du grand nombre de règles impliquées, c'est pourquoi dans le chapitre 4, nous proposons une analyse des théories induites dans les sections qui suivent.

[4] Dans leur formulation du problème d'apprentissage, les descriptions ne sont pas des complexes (en logique attribut-valeur), elles permettent en effet d'exprimer des relations entre objets.



(a) **résultat d'un algorithme de type "diviser pour régner"**

(b) **résultat d'un algorithme de type "séparer pour régner"**

FIG. 3.8 – Schématisation, dans l'espace des versions de Mitchell (voir chapitre 1), de la relation de couverture pour une théorie induite avec un algorithme de type "diviser pour régner", celle-ci partitionne alors l'espace des exemples $\mathcal{E}_E$; et une théorie induite avec un algorithme de type "séparer pour régner" où un exemple peut être couvert par plusieurs règles.

3.4 Découverte de motifs d'arbre d'exutoires

Nous nous intéressons dans cette section à la découverte de motifs d'arbres d'exutoires pour la caractérisation des classes de transfert (tâche définie en 3.7). L'approche consiste à conserver une structure d'arbres dans les règles afin de faire ressortir des relations entre exutoires pertinentes vis-à-vis du transfert. Un arbre d'exutoire est traité ici comme un arbre où les nœuds sont décrits par un ensemble d'attributs. Un motif d'arbre d'exutoires est une règle représentant un arbre où les nœuds contiennent des sélecteurs sur ces attributs. Des exemples de motifs recherchés sont donnés dans la figure 3.21.

De telles motifs ne peuvent être exprimés qu'à l'aide d'un langage relationnel, c'est-à-dire un langage permettant de décrire les relations entre objets ; dans notre cas, les objets sont les nœuds d'un arbre. Si la relation de couverture est intuitive dans le cadre d'une logique attribut-valeur, elle peut être complexe dans une logique relationnelle. L'exemple 3.9 a pour but de montrer la relation de couverture qui nous intéresse pour les motifs d'arbre d'exutoires ; ces derniers peuvent être vus comme des sous-arbres.

Exemple 3.9 (couverture des motifs d'arbre d'exutoires).
Nous présentons ici la relation de couverture qui nous intéresse entre un motif d'arbre d'exutoires et les exemples d'apprentissage. Pour simplifier, les exutoires sont ici décrits à l'aide d'un seul attribut quantitatif *Aquant*. L'interprétation du motif de la figure 3.10 est : la valeur de *Aquant* pour l'exutoire racine doit être supérieure à 3 et l'exutoire racine doit posséder au moins deux fils, et pour l'un d'eux, la valeur de *Aquant* doit être inférieure à 2. Tout d'abord, nous souhaitons contraindre un motif à contenir un exutoire racine. L'exutoire le plus bas décrit donc nécessairement la racine d'un exemple.

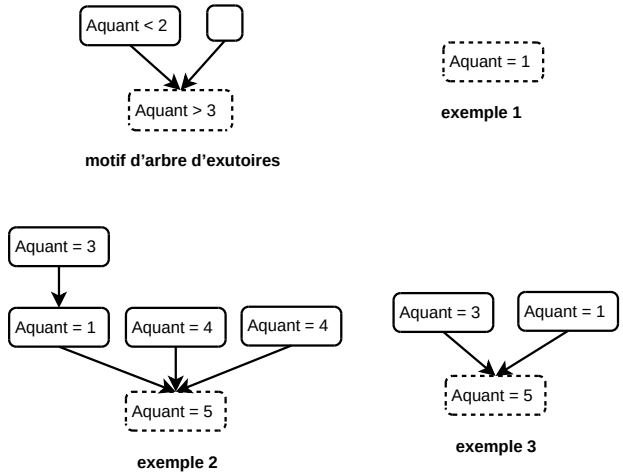


FIG. 3.10 – Les exutoires racine des exemples et du motif sont identifiés par des pointillés. Le motif couvre les exemples d'apprentissage 2 et 3 mais pas l'exemple 1.

Ainsi l'exemple 1, dans lequel la valeur de *Aquant* est 1 pour la racine, n'est pas couvert par le motif qui contraint la valeur de *Aquant*, pour la racine, à être supérieure à 3. Le but est, lors de la visualisation des motifs, d'être certain que l'exutoire le plus bas est un exutoire racine, celui-ci jouant de toute façon un rôle primordial dans le transfert d'herbicides. L'exemple 2 est couvert par le motif, même s'il possède d'autres exutoires que ceux décrits par le motif (notion de sous-arbre *induit*). L'exemple 3 est couvert également, même si l'exutoire fils de la racine respectant la contrainte $Aquant < 2$ est ici représenté à droite ; en effet nous ne souhaitons pas imposer d'ordre sur les fils d'un exutoire (notion d'arbre *non ordonné*).

En bio-informatique, des travaux, tels que ceux décrits dans (Inokuchi *et al.*, 2000) et (Dehaspe *et al.*, 1998), ont consisté à rechercher des motifs dans les composants chimiques. Les composants chimiques sont représentés par des graphes où les nœuds et les arcs sont labellisés, c'est-à-dire qu'ils sont annotés par une valeur d'un attribut de domaine qualitatif. Ces labels représentent des atomes pour les nœuds et des types de liaison entre atomes pour les arcs. Ces travaux ne sont pas adaptés pour notre problème car ils manipulent des graphes et non des arbres et ne répondent pas au problème de la recherche de sélecteurs sur les attributs décrivant les nœuds. Les travaux de Ferré et King (Ferré & King, 2005) suggèrent d'utiliser les logiques dédiées (en anglais : *domain-specific logics*) afin d'adapter la recherche de motifs aux logiques décrivant les données. Dans notre cas, où les données sont des arbres d'exutoires, il serait nécessaire de définir un foncteur logique d'arbre pour la structure primaire des arbres d'exutoires et un foncteur logique attribut-valeur pour les nœuds des arbres.

Nous avons fait le choix d'une recherche descendante dans l'espace des hypothèses en Programmation Logique Inductive (voir sous-section 1.2.5.1) ; les hypothèses étant ici des motifs d'arbre d'exutoires. La logique des prédicats est adaptée pour exprimer

ce type de motifs.

Les notions utiles pour le concept de sous-arbres sont détaillées en sous-section 3.4.1. Avant de donner l'opérateur de spécialisation de clauses pour la recherche de motifs d'arbres d'exutoire (sous-section 3.4.3), nous détaillons les langages des exemples $\mathcal{L}_E$ et des hypothèses $\mathcal{L}_H$ utilisés (sous-section 3.4.2). L'opérateur de spécialisation et la recherche descendante sont mises en œuvre au sein du système Aleph (sous-section 3.4.4).

3.4.1 Notions de sous-arbres

La *taille* d'un arbre est le nombre de nœuds présents dans l'arbre. Un nœud n_p est un *descendant*[5] d'un nœud n_1 si, pour tout i de 1 à $p-1$, n_{i+1} est un fils de n_i. Un *arbre labellisé* est un arbre dont les nœuds ont un label ; par exemple, l'arbre T dans la figure 3.11.

Il existe de nombreux travaux sur la recherche dans les arbres labellisés dont (Chi *et al.*, 2005) constitue une vue d'ensemble. Dans ces travaux, il s'agit de rechercher des sous-arbres fréquents dans une base de données constituée d'arbres labellisés. Trois types de sous-arbres sont distingués (figure 3.11) :

- les sous-arbres exacts (en anglais *bottom up subtrees*). Un sous-arbre exact (a) de T peut être obtenu en conservant un nœud de T et tous ses descendants dans T.
- les sous-arbres induits (en anglais *induced subtrees*). Un sous-arbre induit (b) de T peut être obtenu en enlevant, à plusieurs reprises une feuille de T (ou sa racine si celle ci n'a qu'un fils).
- les sous-arbres cachés (en anglais *embedded subtrees*). Un sous-arbre caché (c) de T doit conserver la relation de descendance entre deux nœuds de T (certains nœuds peuvent être supprimés). A noter que l'ensemble des arcs dans un sous-arbre caché de T n'est pas nécessairement un sous-ensemble des arcs de T.

Un sous-arbre exact de T est un sous-arbre induit de T qui est lui même sous-arbre caché de T.

Les algorithmes décrits dans (Chi *et al.*, 2005) peuvent être directement appliqués à la recherche de sous-arbres non labellisés. Un arbre non labellisé (par exemple les arbres de la figure 3.12) peut être vu comme un arbre labellisé où il n'existe qu'un label possible pour les nœuds. Il existe cependant des algorithmes spécifiques à la recherche dans les arbres non labellisés (Valiente, 2002).

On distingue également les arbres *ordonnés* des arbres *non ordonnés*. Un arbre non ordonné n'impose pas d'ordre sur les fils d'un nœud et donc, à un arbre non ordonné correspondent plusieurs arbres ordonnés dits *isomorphiques*. Par exemple, les arbres (d) et (e) de la figure 3.12 sont deux arbres ordonnés distincts mais ce sont deux arbres non ordonnés isomorphiques (il suffit d'inverser les deux fils de la racine les plus à gauche pour obtenir l'arbre). Un arbre non ordonné *canonique* est un unique représentant d'un ensemble d'arbre non ordonnés isomorphiques. Nakano et Uno (Nakano & Uno, 2003) proposent une définition d'arbre non ordonné canonique et présentent un algorithme

[5]Dans le cadre des arbres d'exutoires, on note que la relation *en amont* (voir définition 2.11) est équivalente à la relation *descendant*.

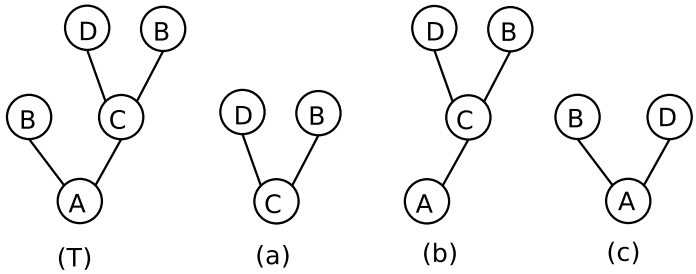


FIG. 3.11 – (T) : arbre labellisé, (a) : sous-arbre exact de T, (b) : sous-arbre induit de T, (c) : sous-arbre caché de T

.

pour les énumérer. L'énumération consiste à définir le procédé qui permet la génération d'un arbre de taille $p+1$ à partir d'un arbre de taille p. L'objectif d'énumérer uniquement les arbres non ordonnés canoniques est d'éviter une redondance importante dans la génération des arbres (voir l'exemple 3.13). Au cours de l'énumération, Nakano et Uno identifient dans un arbre de taille p, les nœuds pour lesquels l'ajout d'un fils permet la génération d'un arbre non ordonné canonique de taille $p+1$.

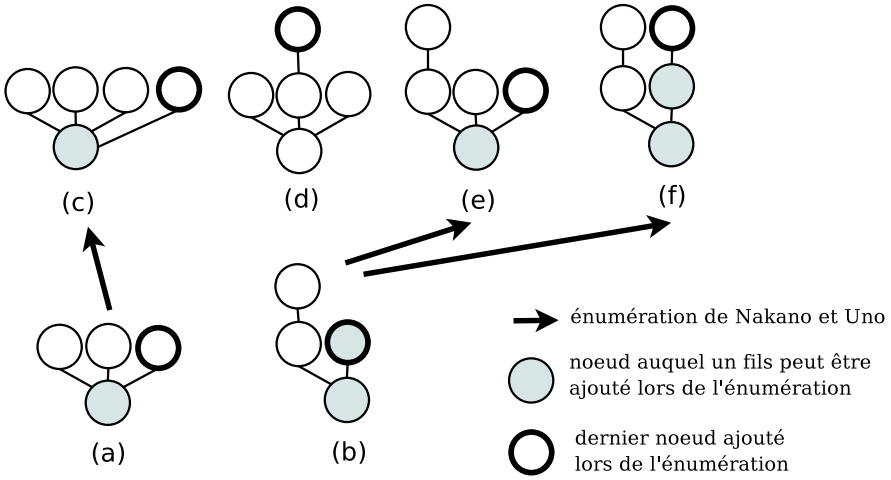


FIG. 3.12 – Les arbres non ordonnés (d) et (e) sont isomorphiques. Seul (e) qui est canonique dans l'ensemble $\{(d),(e)\}$, selon Nakano et Uno (Nakano & Uno, 2003), est généré lors de leur énumération.

Exemple 3.13 (énumération d'arbres).
La figure 3.12 présente deux arbres de taille 3 non ordonnés canoniques d'après (Nakano & Uno, 2003), et la génération, à partir de ces deux arbres, des arbres de taille 4 non ordonnés canoniques. Par exemple, les arbres (d) et (e) sont isomorphiques et l'arbre

canonique est (e), ce dernier est donc généré mais pas (d). Considérons pour l'exemple trois énumérations d'arbres : une pour les arbres non ordonnés canoniques, une pour les arbres ordonnés et la dernière, naïve, qui consiste à ajouter un fils à n'importe quel nœud de l'arbre. Si on s'intéresse aux arbres de taille 9, l'énumération naïve génère 6720 arbres, celle des arbres ordonnés génère 766 arbres et celle des arbres non ordonnés canoniques (telle que proposée par Nakano et Uno) génère 115 arbres. Si l'on s'intéresse aux arbres non ordonnés canoniques, l'énumération naïve génère donc, parmi les 6720 arbres, uniquement 115 arbres différents de taille 9.

L'opérateur de spécialisation de clauses pour la recherche de motifs d'arbre d'exutoires (sous-section 3.4.3) repose entre autres sur l'énumération des arbres non ordonnés canoniques et non labellisés de Nakano et Uno. Les motifs que nous recherchons sont des sous-arbres induits.

3.4.2 Représentation des données d'apprentissage

Afin d'exprimer les relations entre exutoires pour la description d'un exemple de la base d'apprentissage (voir sous-section 3.3.2), une représentation en logique des prédicats est adaptée ; nous nous basons ici sur le langage Prolog (voir annexe A) et un apprentissage de règles par Programmation Logique Inductive (PLI). Mettre en œuvre un tel apprentissage consiste en premier lieu à définir les langages des exemples et des hypothèses. En PLI, mettre en œuvre un tel apprentissage nécessite de déterminer les littéraux positifs qui identifient les exemples dans les ensembles $E^{\oplus}$ et $E^{\ominus}$, ainsi que le langage de la théorie du domaine T et celui des hypothèses $\mathcal{L}_H$. Nous rappelons en effet que nous restreignons les ensembles $E^{\oplus}$ et $E^{\ominus}$ à des ensembles de littéraux positifs (voir section 1.2.1).

Pour simplifier l'écriture, nous introduisons le prédicat $classe/1$ qui est égal au prédicat $transfert\_important/1$ (respectivement $transfert\_faible/1$) dans le cas où l'on cherche à induire une théorie caractéristique de la classe $transfert\_important$ (respectivement $transfert\_faible$). Le littéral $classe(ex)$ désigne donc la classe de l'exemple d'identifiant ex.

La description d'un exemple, c'est à dire l'arbre d'exutoires correspondant, est donnée dans la théorie du domaine T. Chaque exutoire est décrit par des *attributs d'exutoire*.

Définition 3.14 (Attribut d'exutoire).
Un attribut d'exutoire est un attribut qui décrit un exutoire de parcelle. Il peut être qualitatif ou quantitatif. Dans notre représentation, le nom d'un attribut d'exutoire relatif aux données décisionnelles est préfixé par "itk" et celui d'un attribut d'exutoire relatif aux données d'occupation du sol ou topologiques est préfixé par "os".

Dans la théorie du domaine, nous introduisons 3 prédicats pour le langage des exemples[6] :

– $racine(Ex, Exu)$: Exu est l'exutoire racine de l'exemple Ex ;

[6] les termes commençant par une majuscule représente des variables (voir annexe A).

Nom de l'attribut	Type	Description de l'attribut
os_surface	quantitatif	surface de l'exutoire
os_mais	qualitatif	est à *vrai* si le sol est occupé en maïs
os_dispositif_tampon	qualitatif	est à *vrai* si l'exutoire possède un dispositif tampon
os_pente	qualitatif	la pente locale à l'exutoire (nulle, faible ou importante)
os_mrt	quantitatif	un indice topographique
itk_type	qualitatif	type d'itinéraire technique employé (tout_en_post ou pre_puis_post)
itk_pression	quantitatif	quantité totale de pesticide appliqué sur l'exutoire

TAB. 3.15 – Liste des attributs d'exutoire utilisés pour la découverte de motifs d'arbre, les détails sur ces attributs sont donnés en annexe B et dans le chapitre 2.

- $valeur(Ex, Exu, Attr, V)$: V est la valeur de l'attribut d'exutoire $Attr$ pour l'exutoire Exu de l'exemple Ex, l'ensemble des attributs d'exutoires est listé dans le tableau 3.15 ;
- $fils(Ex, Exu_1, Exu_2)$: l'exutoire Exu_2 est un fils (définition 2.11) de l'exutoire Exu_1.

Exemple 3.16 (exemple d'apprentissage pour l'induction des motifs).
L'ensemble de faits suivants forme la description d'un exemple ex de la base d'apprentissage. Cette description est donnée dans la théorie du domaine T.

```
racine(ex1,exu0).
valeur(ex1,exu0,os_mais,faux).
valeur(ex1,exu0,os_surface,0.4).
valeur(ex1,exu0,os_bande_enherbee,faux).
valeur(ex1,exu0,os_pente,forte).
fils(ex1,exu0,exu1).
valeur(ex1,exu1,os_mais,vrai).
valeur(ex1,exu1,os_surface,0.84).
valeur(ex1,exu1,os_bande_enherbee,faux).
valeur(ex1,exu1,os_pente,forte).
valeur(ex1,exu1,itk_type,tout_en_post).
valeur(ex1,exu1,itk_pression,22.0).
```

Cet exemple, identifié par $ex1$ représente un arbre d'exutoires à deux exutoires $exu0$ et $exu1$; $exu1$ est un exutoire de parcelle cultivée en maïs et est un fils de $exu0$.

Pour spécifier le langage des hypothèses $\mathcal{L}_H$, de nouveaux prédicats sont définis, il s'agit des comparateurs $<=$, $>=$ et $=$ et d'un prédicat d'introduction d'un nouvel exutoire dans le motif :

- $inf\_egal(X, seuil)$: la valeur X est inférieure ou égale à $seuil$;
- $sup\_egal(X, seuil)$: la valeur X est supérieure ou égale à $seuil$;
- $egal(X, cste)$: la valeur X est égale à la constante $cste$;
- $intro\_noeud(Ex, Exu_0, ListExu, Exu_1)$: pour l'exemple Ex, Exu_1 est un exutoire fils de l'exutoire Exu_0 et il est différent de tout exutoire de la liste de variables $ListExu$; cette liste répertorie en fait les nœuds fils de $Exu0$ déjà présents dans le motif. L'exemple 3.17 montre l'intérêt de l'utilisation de ce prédicat, à la place du prédicat $fils/2$.

Le langage des hypothèses repose également sur les prédicats $racine/1$, $valeur/4$ et $classe/1$ définis précédemment.

Exemple 3.17 (intérêt de l'utilisation du prédicat *intro_noeud*/4).
Soit la clause suivante (non incluse dans $\mathcal{L}_H$) :

```
classe(Ex) :-racine(Ex,Exu0),fils(Exu0,Exu1),fils(Exu0,Exu2).
```

On peut penser qu'il s'agit d'un motif d'arbre d'exutoires constitué d'un exutoire racine $Exu0$ qui possède deux fils $Exu1$ et $Exu2$. Mais $Exu0$, $Exu1$ et $Exu2$ sont trois variables et non des constantes. Un arbre d'exutoires constitué d'une racine $Exu0$ ne possédant qu'un fils est couvert par la clause ; en effet, les variables $Exu1$ et $Exu2$ peuvent être unifiées au même exutoire. Pour éviter ce problème, le prédicat $intro\_noeud/4$ est préféré à $fils/2$:

```
classe(Ex) :-racine(Ex,Exu0),intro_noeud(Ex,Exu0,[Exu0],Exu1),
             intro_noeud(Ex,Exu0,[Exu0,Exu1],Exu2).
```

Ainsi, $Exu1$ et $Exu2$ représentent nécessairement deux exutoires différents. Une visualisation graphique (représentant deux exutoires différents) peut être envisagée.

Dans notre adaptation d'un algorithme de PLI à l'apprentissage de motifs d'arbres d'exutoires, le langage des hypothèses $\mathcal{L}_H$ est l'ensemble des clauses dont la tête représente une classe de transfert d'herbicides et le corps, un motif d'arbre d'exutoires. Ainsi, la deuxième clause de l'exemple 3.17 est une clause du langage $\mathcal{L}_H$.

3.4.3 Opérateur de spécialisation de clauses pour l'induction de motifs d'arbre d'exutoires

Pour la réalisation d'une recherche descendante de motifs d'arbres d'exutoires (sous-section 1.2.5.1), il faut déterminer un opérateur de spécialisation permettant de construire de nouveaux motifs d'arbres, plus spécifiques, au sens de la relation $\succeq_\theta$ de θ-subsomption, à partir d'un motif déjà construit. Nous décomposons cet opérateur en deux opérateurs, chacun permettant de générer de nouvelles clauses plus spécifiques à évaluer sur l'ensemble d'apprentissage. Voici les deux opérateurs utilisés :

spec1 : introduction dans le motif d'arbre d'un nouvel exutoire. Seuls les arbres canoniques, selon (Nakano & Uno, 2003), doivent être générés. Pour cela l'ensemble P des exutoires pour lesquels l'ajout d'un fils permet la construction d'un arbre non ordonné non labellisé canonique de taille supérieure est calculé (voir sous-section 3.4.1) ; le nouvel exutoire est alors introduit en tant que fils d'un exutoire de P.

Cette spécialisation de clause correspond à une augmentation du nombre d'exutoires de l'arbre. Une énumération des arbres non ordonnés a pour but d'éviter une redondance importante dans la construction des clauses à tester (voir l'exemple 3.13). L'opérateur *spec*1 repose sur l'ajout d'un littéral de prédicat $intro\_noeud/4$ à la clause.

spec2 : ajout d'une contrainte sur la valeur d'un attribut d'exutoire, pour le dernier exutoire introduit dans l'arbre. La constante utilisée pour la comparaison est évaluée de manière paresseuse (de l'anglais *lazy evaluation*). C'est à dire qu'elle est calculée en fonction des exemples couverts par une clause intermédiaire où la constante est remplacée par une variable (voir l'exemple 3.20). L'opérateur *spec*2 repose sur l'ajout à la clause d'un littéral de prédicat $valeur/4$ et d'un littéral de comparaison ($inf\_egal/2$, $sup\_egal/2$ ou $egal/2$).

L'opérateur ainsi défini (composé de spec1 et spec2) est un opérateur classique de spécialisation de clause puisqu'il repose sur l'ajout de prédicats à une clause (voir l'exemple 3.18). Ainsi, une clause θ-subsume bien ses spécialisations. Des exemples de motifs d'arbres générés à l'aide de ces deux opérateurs sont donnés dans l'exemple 3.18. Chaque motif généré est évalué à l'aide d'un test de couverture sur les exemples d'apprentissage.

Exemple 3.18 (un motif d'arbre d'exutoire et sa spécialisation).
La figure 3.19 présente un motif d'arbre d'exutoire (a) et quelques unes de ces spécialisations (b), (c), (d) et (e) par l'opérateur défini ci-dessus. Le motif d'arbre d'exutoires (a) est représenté par le corps de la clause suivante :

```
classe(E) :-racine(E,Exu1),intro_noeud(E,Exu1,[],Exu2).
```

La spécialisation (b) du motif (a) par l'ajout d'un nœud s'écrit :

```
classe(E) :-racine(E,Exu1),intro_noeud(E,Exu1,[],Exu2),
            intro_noeud(E,Exu2,[],Exu3).
```

La spécialisation (c) du motif (a) par l'ajout d'un nœud s'écrit :

```
classe(E) :-racine(E,Exu1),intro_noeud(E,Exu1,[],Exu2),
            intro_noeud(E,Exu1,[Exu2],Exu3).
```

Dans cette spécialisation, il faut s'assurer que l'exutoire identifié par la variable $Exu3$ est bien différent de l'exutoire identifié par la variable $Exu2$, ces deux exutoires étant fils de l'exutoire identifié par la variable $Exu1$.

La spécialisation (d) du motif (a) par l'ajout d'un sélecteur sur l'attribut $os\_dispositif\_tampon$ de l'exutoire $Exu2$ s'écrit :

```
classe(E) :-racine(E,Exu1),intro_noeud(E,Exu1,[],Exu2),
            valeur(E,Exu2,os_dispositif_tampon,V),egal(V,vrai).
```

La spécialisation (e) du motif (a) par l'ajout d'un sélecteur sur l'attribut $os\_surface$ de l'exutoire $Exu2$ s'écrit :

```
classe(E) :-racine(E,Exu1),intro_noeud(E,Exu1,[],Exu2),
            valeur(E,Exu2,os_surface,V),sup_egal(V,0.4).
```

A partir du motif d'arbre d'exutoires (a) de taille 2, tous les arbres canoniques de taille 3 générés sont explicités dans cet exemple. Par contre, tous les motifs issus de l'ajout de sélecteurs sur un attribut décrivant le dernier exutoire ajouté ne sont pas présents.

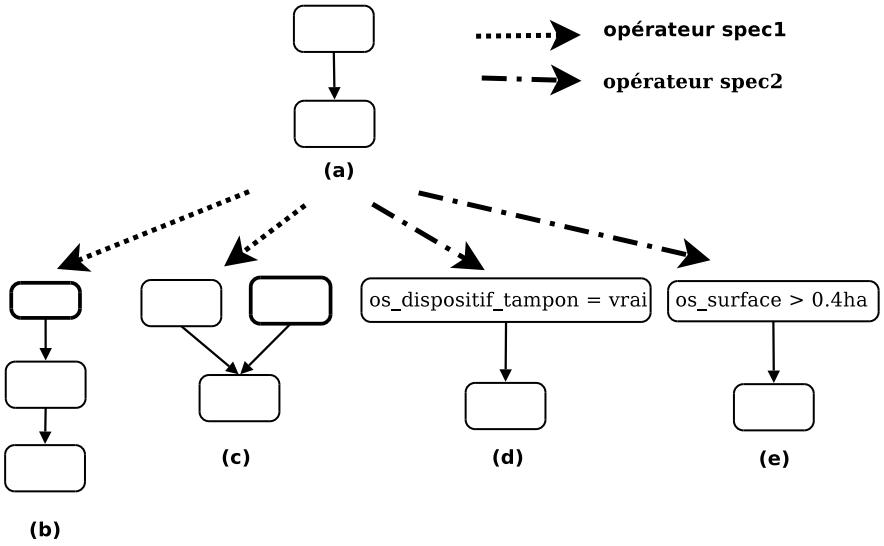


FIG. 3.19 – Exemples de motifs d'arbres d'exutoires générés (b, c, d, e) à l'aide des opérateurs de spécialisation de clauses **spec1** et **spec2**, et à partir d'une clause représentant un arbre d'exutoires à deux exutoires (a) sans sélecteurs sur les attributs aux nœuds.

3.4.4 Mise en œuvre au sein du système Aleph

L'opérateur de spécialisation pour la recherche de motifs et la recherche descendante en elle même sont mis en œuvre au sein du système Aleph (sous-section 1.2.6.2). Le prédicat $refine/2$, permettant de définir les spécialisations d'une clause a été implémenté en s'appuyant sur les opérateurs de spécialisation **spec1** et **spec2**.

L'étape 2 (dite de réduction) pour la génération d'une hypothèse au sein du système Aleph (voir algorithme 1.26) consiste en l'élaboration de motifs qui, représentés par des clauses, doivent être contenus dans la *bottom clause*. La *bottom clause* est finalement une base de littéraux qui peuvent constituer l'hypothèse. Elle est élaborée lors de l'étape de saturation d'un exemple d'apprentissage non encore couvert (étape 1).

Dans notre cas, les littéraux de prédicat $intro\_noeud/4$ ne sont déterminés que lors de l'élaboration du motif ; en effet, le troisième argument de ce prédicat repose sur les variables d'exutoires préalablement introduites dans le motif à l'aide de ce même prédicat (voir l'exemple 3.17). Ces variables ne sont donc pas connues au moment de la phase de construction de la *bottom clause*, où aucun motif n'est généré. La construction de la *bottom clause* est donc rendue impossible par l'utilisation du prédicat $intro\_noeud/4$.

Les prédicats $inf\_egal/2$, $sup\_egal/2$ et $egal/2$ sont évalués de manière paresseuse (Srinivasan & Camacho, 1999). Les deuxièmes arguments de ces prédicats sont des constantes du domaine de l'attribut considéré. La génération exhaustive de toutes les constantes n'est pas envisageable, surtout dans le cas d'attributs quantitatifs, il s'agit alors de déterminer une constante qui permette d'élaborer un motif significatif de la

classe que l'on cherche à caractériser ; c'est ce procédé que l'on appelle l'*évaluation paresseuse* de prédicat (voir l'exemple 3.20).

Exemple 3.20 (l'évaluation paresseuse de prédicat en PLI).
Soit la clause $C1$ suivante, où *cste* peut prendre la valeur *vrai* ou $faux$:
`classe(E) :-racine(E,Exu1),valeur(E,Exu2,os_mais,V),egal(V,cste).`
Le prédicat à évaluer de manière paresseuse est le prédicat $egal/2$, il s'agit en fait d'assigner à *cste* la valeur *vrai* ou $faux$. L'évaluation paresseuse telle que mise en œuvre dans (Srinivasan & Camacho, 1999) commence par construire la clause suivante $C2$ où une variable X est introduite là où devrait figurer la constante :
`classe(E) :-racine(E,Exu1),valeur(E,Exu2,os_mais,V),egal(V,X).`
Ensuite le procédé stocke les exemples d'apprentissages couverts par cette clause et surtout les constantes, pour chacun des exemples, qui sont unifiées à X lors des preuves Prolog de couverture des exemples. Afin de construire la clause la plus significative pour la classe à caractériser, nous assignons à *cste* la valeur qui est la plus souvent unifiée à X lors des preuves de couverture des exemples de classe à caractériser. Prenons trois exemples $ex1$, $ex2$ et $ex3$ tels $ex1$ et $ex2$ soient de classe $transfert\_important$ et $ex3$ de classe $transfert\_faible$. Considérons qu'une partie de leur description dans T soit :
`racine(ex1,exu0).`
`valeur(ex1,exu0,os_mais,vrai).`
`racine(ex2,exu1).`
`valeur(ex2,exu1,os_mais,vrai).`
`racine(ex3,exu1).`
`valeur(ex3,exu1,os_mais,faux).`
Alors, pour caractériser la classe $transfert\_important$, le procédé d'évaluation paresseuse substitue X à la constante *vrai* dans la clause $C2$ pour générer $C1$.

On note qu'un algorithme de recherche descendante utilisant ce procédé n'est pas un algorithme de type "générer et tester" (sous-section 1.2.5.1) mais un algorithme de type "guidé par les exemples" ; les exemples sont en effet utilisés dans la détermination de la constante et donc dans l'élaboration du motif.

Enfin, lors de l'étape de réduction, nous effectuons une recherche descendante par faisceau où la spécialisation de clause est définie par le prédicat $refine/2$. La recherche par faisceau est employée au sein du système CN2 présenté en sous-section 1.1.2.

Afin d'induire une théorie caractéristique d'une classe d'apprentissage (voir définition 3.7), une stratégie de type "séparer pour régner" (voir l'algorithme 1.9) est adoptée pour l'apprentissage des règles. Pour générer une règle, nous optons pour une recherche descendante par faiseau. La recherche par faisceau est utilisée au sein du système CN2 (voir l'algorithme 1.8) dans le cadre d'une logique attribut-valeur. Cependant, pour la recherche de motifs d'arbre d'exutoires, la règle la plus générale est la règle " `class(Ex) :-racine(Ex,Exu1)` ", c'est-à-dire un arbre d'exutoire ne possédant qu'un exutoire racine. De plus, la spécialisation d'une règle du faisceau s'effectue à l'aide de l'opérateur de spécialisation proposé en sous-section 3.4.3 et les techniques d'élagage liées à une recherche descendante (sous-section 1.2.5.4) sont mises en œuvre.

3.4.5 Quelques résultats

En utilisant la base d'apprentissage définie dans la sous-section 3.3.2, 28 motifs d'arbres d'exutoires ont été générés, répartis en 14 motifs caractéristiques de la classe $transfert\_faible$ et 14 autres de la classe $transfert\_important$. L'ensemble de ces motifs est donné en annexe C.

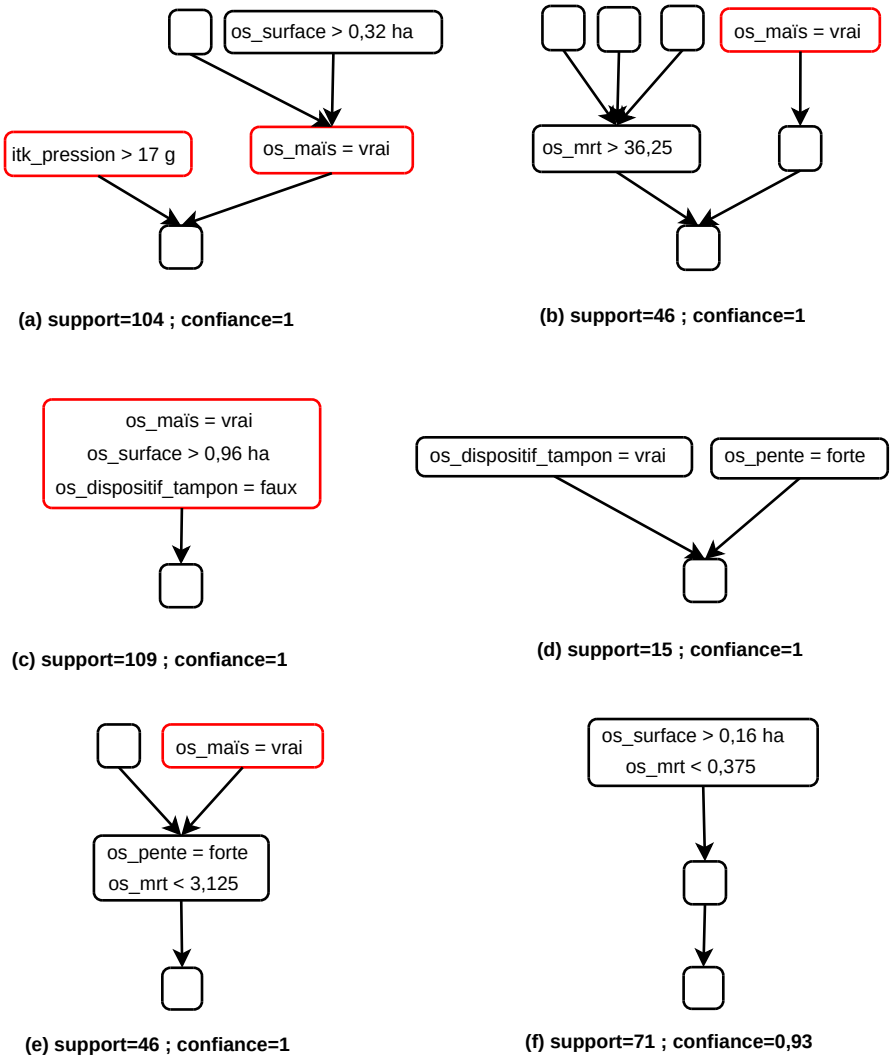


FIG. 3.21 – Six motifs d'arbres d'exutoires induits : trois sont caractéristiques d'un transfert important (a,b et c) et trois sont caractéristiques d'un transfert faible (d, e et f). Le support d'un motif est le nombre d'exemples d'apprentissages couverts par le motif et la confiance (ou précision) est la proportion de ces exemples qui sont de la même classe que celle du motif. Pour faciliter la visualisation, les exutoires nécessairement en maïs sont en rouge.

De manière générale, nous nous attendions à caractériser un transfert d'herbicides important par la présence dans les motifs d'exutoires de parcelle maïs proches du cours d'eau, ce qui a été le cas. Sur 8 motifs caractéristiques d'un transfert important, au moins un exutoire proche du cours d'eau (à une profondeur 0 ou 1) est nécessairement en occupation maïs (motifs a et c de la figure 3.21).

Le motif c représente les arbres d'exutoires qui ont un exutoire de parcelle maïs, non équipé d'une bande enherbée et de surface importante se déversant directement dans l'exutoire racine ; ce qui représente une situation typique de transfert important. Le motif a représente les arbres dont la racine possède deux exutoires de parcelle maïs, pour lesquels une quantité minimale de $17g$ d'herbicides est appliquée ou dont la surface contributive (c'est-à-dire la surface en amont) n'est pas négligeable. Le motif b représente un arbre dont une première partie, d'indice topographique fort et alimentée par de nombreux exutoires, contribue probablement à un apport d'eau important sur la racine de l'arbre. La deuxième partie de ce motif possède un exutoire de parcelle maïs à la profondeur 2, donc plutôt proche de la rivière. On note aussi, à l'aide du sélecteur $os\_mrt > 36,25$, que les arbres couverts par ce motif sont plutôt des arbres de bas de versant, donc au niveau de surfaces plus souvent saturées.

De nombreux motifs (9 sur 15) caractérisent un arbre de classe $transfert\_important$ par la présence d'un exutoire de parcelle maïs à la profondeur 1 ou 2 et de nombreux exutoires à proximité du cours d'eau ; ce sont des arbres en forme d'entonnoir. Le nombre important d'exutoires peut expliquer une saturation en eau du sol et faciliter le transfert à proximité du cours d'eau.

Des motifs caractéristiques d'un transfert faible introduisent des sélecteurs sur la présence d'un dispositif tampon sur un exutoire proche du cours d'eau. Par exemple, le motif d représente les arbres dont la racine possède un exutoire fils équipé d'un dispositif tampon et un autre exutoire fils sur lequel la pente est forte. En effet, au sein du modèle, la pente n'est pas intégrée dans le calcul des quantités d'herbicides transférées d'un exutoire vers son exutoire père. Les résultats de simulation montrent qu'une pente forte ne facilite pas le transfert : les exutoires de pente forte sont plutôt en haut de bassin versant et correspondent en général à des endroits du bassin versant où la nappe est éloignée de la surface du sol, ce qui est aussi le cas généralement lorsque l'indice topographique est faible (attribut os_mrt). Pour interpréter le motif e, nous faisons le même constat : l'exutoire de pente forte, situé en haut de bassin versant, joue un rôle tampon sur le transfert d'herbicides provenant de l'amont.

Le motif f caractéristique d'un transfert faible est un motif linéaire introduisant des sélecteurs sur des données de topologie (type de motifs plutôt présent pour la classe $transfert\_faible$) où l'exutoire de profondeur 2 est de taille importante et se situe en haut de bassin versant.

Les motifs générés, pour les deux classes, sont de manière générale des motifs introduisant des sélecteurs sur des données topologiques ou d'occupation du sol. L'indice topographique (équation 2.3) qui combine la pente et la surface drainée est très présent dans les résultats. Concernant les décisions liées aux itinéraires techniques, seuls des sélecteurs sur les quantités totales de pesticides appliquées (attribut $itk\_pression$) sont parfois présents dans les motifs résultant. Au contraire, les résultats issus de l'appren-

tissage de règles attributs valeurs montrent des règles composées de sélecteurs sur les données d'itinéraires techniques.

3.5 Découverte de règles attribut-valeur pour les arbres d'exutoires

Dans cette section, l'approche proposée pour caractériser une classe de transfert (voir définition 3.7) consiste en la définition d'*attributs agrégats* qui résument l'information contenue dans un arbre d'exutoires. Par exemple, la somme des surfaces des exutoires dans un arbre d'exutoires est un attribut agrégat.

Définition 3.22 (Attribut agrégat).
Un attribut agrégat est un attribut qui permet la description d'un exutoire et de ses exutoires en amont. Pour un exutoire racine, un attribut agrégat synthétise l'information contenue dans tout l'arbre d'exutoires. Dans notre représentation le nom d'un attribut agrégat est préfixé par "agr".

L'objectif est de décrire les exemples d'apprentissage dans une logique attribut-valeur (voir section 1.1). Dans cette logique, les systèmes d'induction sont plus efficaces en terme de temps de calcul. Avant la présentation de quelques règles données en sous-section 3.5.2, nous déterminons l'ensemble des attributs agrégats utilisés (sous-section 3.5.1).

3.5.1 Sélection de l'ensemble des attributs agrégats

Les attributs agrégats peuvent synthétiser des informations tant sur les itinéraires techniques de désherbage (ITK) que sur les données topologiques ou d'occupation spatiale décrivant les exutoires de parcelle. Dans cette section, ils sont calculés pour les exutoires racine. Ainsi, il peut être intéressant de connaître la surface de sol cultivée en maïs sur la totalité d'un arbre d'exutoires. La liste des attributs agrégats sélectionnés de manière experte est donnée dans le tableau 3.23. Un exemple de description d'un arbre d'exutoires à l'aide de ces attributs est illustré dans la figure 3.24.

Les attributs agrégats utilisés ont été sélectionnés parmi une trentaine d'attributs en fonction de leur pertinence et de l'interprétabilité des règles obtenues. Leur définition s'est faite de manière incrémentale en testant, à plusieurs reprises, de nouveaux attributs et en analysant les règles obtenues. Le système CN2 (sous-section 1.1.2) est le système utilisé pour l'apprentissage de règles en logique attribut-valeur. En résultat, on obtient des informations sur des caractéristiques générales des arbres d'exutoires dans les cas de transfert important et faible.

3.5.2 Quelques résultats

En utilisant la base d'apprentissage définie dans la sous-section 3.3.2, 33 règles ont été générées, dont 17 sont caractéristiques de la classe $transfert\_faible$ et 16 de la classe $transfert\_important$. L'ensemble de ces règles est donné en annexe C.

Nom de l'attribut	**Type**	**Description de l'attribut**
agr_pression	quantitatif	quantité totale de matière active utilisée
agr_rapport_risque_faible	quantitatif	pourcentage de matière active utilisée à risque faible
agr_rapport_risque_fort	quantitatif	pourcentage de matière active utilisée à risque fort
agr_rapport_prelevee	quantitatif	pourcentage de matière active utilisée au stade de prélevée
agr_surface	quantitatif	surface totale de l'arbre d'exutoires
agr_rapport_surf_mais	quantitatif	pourcentage de surface cultivée en maïs
agr_surf_max_mais	quantitatif	surface de l'exutoire de parcelle maïs le plus important
agr_rapport_disp_tampon	quantitatif	pourcentage de surface utilisée comme dispositif tampon
agr_max_profondeur	quantitatif	profondeur maximale de l'arbre
agr_topo_forme_denivele	qualitatif	forme de l'arbre en fonction des pentes (concave, convexe, pentue ou plate)
agr_topo_forme_largeur	qualitatif	forme de l'arbre en fonction de la répartition des exutoires selon leur profondeur (u, v, i ou isole). Un arbre possédant beaucoup d'exutoires à proximité du cours d'eau est de type u (annexe B)

TAB. 3.23 – Liste des attributs agrégats utilisés pour la découverte de règles attribut-valeur, les détails sur ces attributs sont donnés en annexe B.

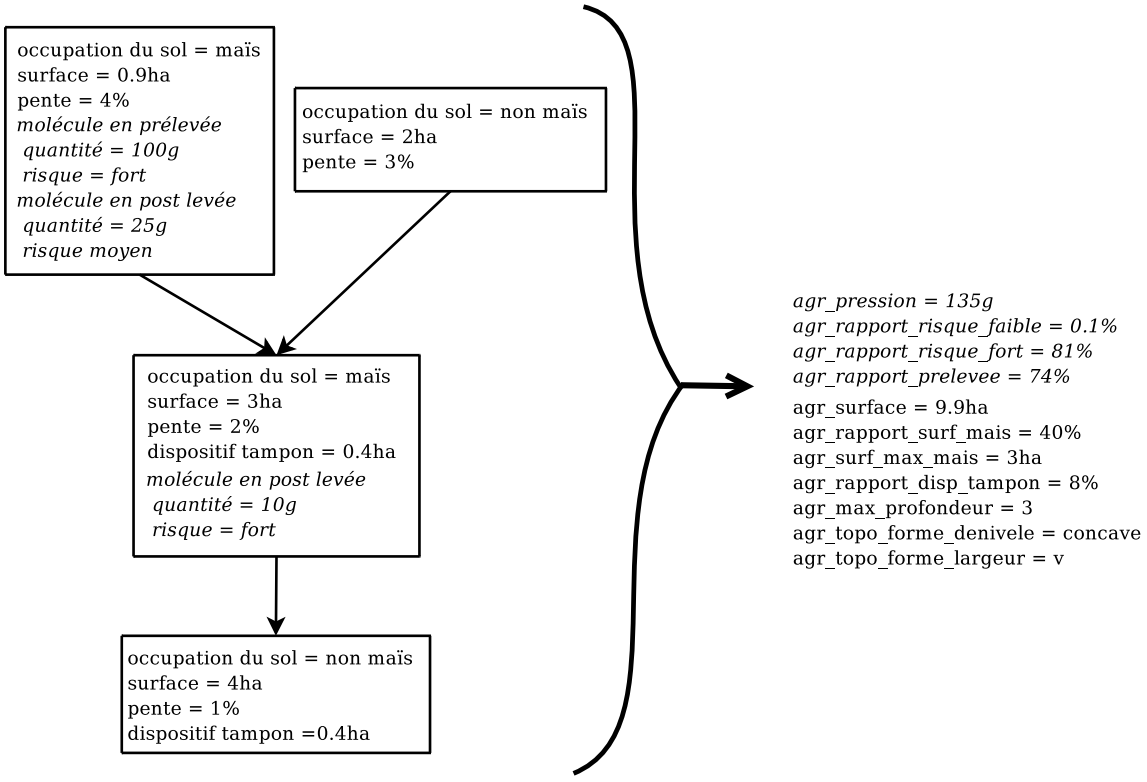


FIG. 3.24 – Les attributs agrégats permettent une description dans une logique attribut-valeur des arbres d'exutoires, c'est à dire des exemples.

Les règles résultant de cet apprentissage introduisent de manière générale un nombre de sélecteurs sur des attributs relatifs à l'itinéraire technique plus important que pour l'apprentissage de motifs d'arbres d'exutoires (section 3.4). Par exemple, la règle suivante prédit un transfert important pour un arbre d'exutoires possédant au moins un exutoire de parcelle maïs de taille importante (au moins 1,54 ha) et l'utilisation de matières actives à risque fort (au moins 50 % de la quantité totale appliquée).

```
SI agr_surf_max_maïs > 1,54 ha
ET agr_rapport_risque_fort > 50 %
ALORS classe = transfert_important [support=125,confiance=1]
```

La non-présence de dispositif tampon sur la surface d'un arbre d'exutoires est aussi un facteur facilitant le transfert. Si la surface de l'arbre est très utilisée pour la culture du maïs (au moins 53,5 %), sur laquelle une quantité non négligeable de matière active est appliquée (au moins 61,5 g), une présence faible de dispositif tampon (au plus 16,5 % de la surface totale de l'arbre) implique un transfert important.

```
SI agr_rapport_disp_tampon < 16,5 %
ET agr_rapport_surf_maïs > 53,5 %
ET agr_pression > 61,5 g
ET agr_topo_forme_denivelé = plate
ALORS classe = transfert_important [support=92,confiance=1]
```

Dans cette règle, le dénivelé sur la surface de l'arbre doit être nul. Comme remarqué précédemment (sous-section 3.4.5), des exutoires de pente forte peuvent jouer un rôle tampon sur le transfert, ce qui peut être expliqué par les choix de modélisation.

Un arbre au taux de transfert faible est souvent caractérisé par une forme linéaire (en i). La règle suivante tend à montrer cela dans le cas d'une utilisation moyenne pour la culture du maïs (entre 10% et 39,5%) de la surface non négligeable de l'arbre (au moins 0,18 ha).

```
SI 10 % < agr_rapport_surf_mais < 39,5 %
ET agr_topo_forme_largeur = i
ET agr_surface > 0,18 ha
ALORS classe = transfert_faible [support=54,confiance=1]
```

La règle suivante, possédant le support le plus important, caractérise un arbre d'exutoires au transfert faible par une surface plutôt faible (entre 0,62 ha et 1,58 ha) et une profondeur maximale moyenne (entre 4,5 et 7,5 exutoires).

```
SI 4,5 < agr_max_profondeur < 7,5
ET 0,62 ha < agr_surface < 1,58 ha
ALORS classe = transfert_faible [support=80,confiance=1]
```

Le type d'ITK, qui définit les fenêtres temporelles d'application des herbicides, ressort moins clairement. Par exemple, la règle suivante, un peu inattendue, caractérise un arbre d'exutoires au transfert faible par un arbre dont les exutoires sont proches du cours d'eau (à une profondeur au plus 3), où la culture du maïs est plutôt importante (entre 35,5 % et 67,5% de la surface) et l'application d'herbicides se fait principalement juste après le semis (au moins 65,5 % de la quantité d'herbicides totale apportée).

```
SI agr_rapport_prelevee > 65,5 %
ET 35,5 % < agr_rapport_surf_mais < 67.5 %
```

```
ET agr_max_profondeur < 3,5
ALORS classe = transfert_faible [support=34,confiance=0,74]
```

Nous nous sommes intéressés à analyser les sélecteurs présents dans les règles afin de dégager la différence entre l'impact attendu des différents facteurs et leur impact d'après les règles obtenues. Par exemple on s'attend à ce qu'une valeur importante de l'attribut *agr_pression* (quantité totale d'herbicides appliquée sur l'arbre) implique un taux de transfert important. On s'attend donc à obtenir des règles pour la classe $transfert\_important$ (respectivement $transfert\_faible$) s'appuyant sur des sélecteurs qui consistent en l'introduction d'une borne minimale sur la valeur de cet attribut. Cette analyse a montré que les deux attributs $agr\_forme\_topo\_denivele$ et $agr\_rapport\_prelevee$ ne se comportent pas comme nous l'attendions.

Pour le premier attribut, lié à la pente de chacun des extoires, l'explication vient certainement des choix de modélisation : la pente n'est en effet pas intégrée dans le calcul des quantités d'herbicides transférées. Pour le second, on peut penser qu'une proportion importante des pesticides appliquées en prélevée, lorsque le couvert végétal est faible, peut faciliter son transfert. De plus, les pesticides appliqués en prélevée comporte généralement un risque plus grand. On remarque plus souvent le comportement inverse, c'est-à-dire qu'une proportion importante de pesticides appliqués en post-levée implique un transfert important. Une explication possible est qu'en début d'année culturale, le sol a drainé peu d'eau et son état structural facilite l'infiltration d'éléments.

3.6 Combinaison de la PLI et l'apprentissage en logique attribut-valeur

L'approche employée lors de la découverte de motifs d'arbres d'exutoires (section 3.4) s'appuie sur une spécialisation de clauses reposant, d'une part, sur l'ajout d'un exutoire fils à un exutoire présent dans le motif et, d'autre part, sur la définition de contrainte sur les valeurs des attributs d'exutoire. L'approche de découverte de règles attribut-valeur (section 3.5) s'appuie quant à elle sur des attributs agrégats, chacun synthétisant une information relative à un type de données dans l'arbre d'exutoires.

Il est possible pourtant de conserver la structure d'arbres d'exutoires tout en calculant pour chaque exutoire les valeurs des attributs agrégats. L'idée est d'utiliser les attributs agrégats comme attributs d'exutoires. C'est cette approche que nous abordons dans cette section. Nous souhaitons utiliser la mise en œuvre de la recherche de motifs d'arbres d'exutoires en décrivant les exutoires en ajoutant les attributs agrégats (voir tableau 3.23) aux attributs d'exutoires (voir tableau 3.15). Ainsi la spécialisation de clauses peut se focaliser, à un nœud donné, sur l'ajout d'un nœuds fils (déploiement de l'arbre), ou sur l'ajout de sélecteurs sur les attributs d'exutoires ou bien sur les attributs agrégats.

Dans l'approche de recherche de motifs, l'opérateur de spécialisation de clauses par ajout d'une contrainte sur la valeur d'un attribut quantitatif consiste à déterminer, de manière exclusive, soit une borne inférieure sur la valeur de l'attribut, soit une borne supérieure. Le système CN2 quant à lui peut construire des sélecteurs symbolisant des

contraintes à la fois sur les bornes inférieures et supérieures (voir les exemples de règles sous-section 3.5.2). Mis à part ces sélecteurs symbolisant une double borne (inférieure et supérieure) sur la valeur d'un attribut, l'approche adoptée dans cette section considère finalement un espace d'hypothèses qui est un sur-ensemble des espaces d'hypothèses considérés dans les approches PLI et logique attribut-valeur.

3.6.1 Quelques résultats

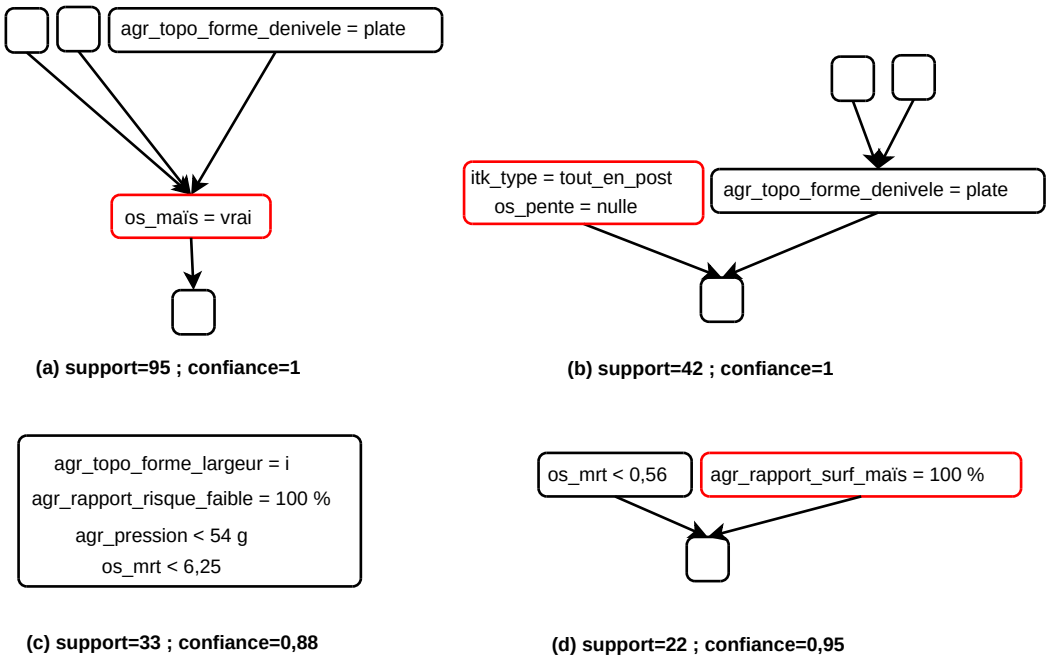


FIG. 3.25 – Les motifs a et b sont caractéristiques d'un transfert important et les motifs c et d d'un transfert faible.

En utilisant la base d'apprentissage définie dans la sous-section 3.3.2, 34 motifs d'arbres d'exutoires ont été générés, dont 18 sont des motifs caractéristiques de la classe $transfert\_faible$ et 16 de la classe $transfert\_important$. L'ensemble de ces motifs est donné en annexe C.

Ces motifs sont des arbres en général très peu développés, 21 motifs sur les 34 appris sont d'ailleurs des arbres à un seul exutoire. Les attributs agrégats ou d'exutoires décrivant les racines des arbres permettent l'introduction d'un ensemble de sélecteurs sur l'exutoire racine suffisamment significatifs de la classe à caractériser. Seulement 4 motifs parmi les 21 motifs à un seul exutoire introduisent des sélecteurs sur les attributs agrégats uniquement. On note que ces motifs sont exprimables dans le langage des hypothèses adopté dans la section 3.5.2 ; un motif très proche de la première règle attribut-valeur présentée a d'ailleurs été généré.

On peut rapprocher les motifs a et b présentés dans la figure 3.25 des motifs a et b présentés dans la figure 3.21. Leurs interprétations peuvent être proches. Ici, pourtant, le sélecteur sur l'attribut $agr\_forme\_topo\_denivele$ présent dans les deux motifs suggère que la surface contributive en eau doit être plate. On a en effet remarqué qu'une pente forte pouvait conduire à un transfert faible dans les résultats de simulation (voir sous-sections 3.4.5 et 3.5.2).

3.7 Résultats d'apprentissage

Dans cette section, les résultats en *classification* de différentes théories induites sont donnés. La classification est le procédé qui consiste à assigner, à l'aide d'une théorie, la classe à un exemple. Les taux de bonnes classifications sont calculés à partir des exemples tests de la base d'apprentissage (voir sous section 3.3.2). La méthode de classification utilisée est celle définie par Clark et Boswell (Clark & Boswell, 1991). La classification d'un exemple ex à partir d'une théorie non ordonnée se fait à l'aide de toutes les règles de la théorie qui couvrent ex (voir sous section 1.1.3) :

- si aucune règle ne couvre ex, alors la classification assigne à ex la classe par défaut qui est celle majoritaire parmi les exemples d'apprentissage ;
- si toutes les règles couvrant ex sont de même classe c alors la classification assigne à ex l'étiquette c ;
- si, enfin, les règles qui couvrent ex sont de différentes classes alors on s'intéresse au support de ces règles afin de déterminer la classe de ex. La classification étiquette ex de la classe majoritaire parmi l'ensemble des exemples d'apprentissages couverts par ces règles.

Les théories auxquelles nous nous intéressons sont les suivantes :

- $\mathcal{MOT}$: l'ensemble des motifs d'arbre d'exutoires issus de l'apprentissage présenté dans la section 3.4 ;
- $\mathcal{AGR}$: l'ensemble des règles attribut-valeur, composé de sélecteurs sur les attributs agrégats, issus de l'apprentissage présenté en section 3.5 ;
- $\mathcal{COMB}$: l'ensemble des motifs d'arbres, combiné à la recherche de sélecteurs sur les attributs agrégats, issus de l'apprentissage présenté en section 3.6.

L'apprentissage des théories s'appuie sur un algorithme de type "séparer pour régner", plus précisément sur une recherche de règles par faisceau (voir sous-section 1.1.3). La taille du faisceau a été fixée à 50, le support minimal à 15, la confiance minimale à 0.6 et l'heuristique utilisée est le $m-estimate$ (voir exemple 1.19). Pour l'apprentissage de motifs d'arbres d'exutoires (théories $\mathcal{MOT}$ et $\mathcal{COMB}$), le nombre maximal de nœuds est fixé à 9.

Les temps d'apprentissage[7], ainsi que les taux de classification et les matrices de confusion pour chacune des théories sont données dans le tableau 3.26. Une matrice de confusion est un tableau de taille 2*2. Les cellules en haut à gauche, en bas à gauche, en haut à droite et en bas à droite représentent respectivement le nombre d'exemples

[7] les expériences ont été effectuées sur un Intel Xeon 3.4GHz.

tests d'étiquette $transfert\_important$ classés $transfert\_important$, celui d'exemples tests d'étiquette $transfert\_important$ classés $transfert\_faible$, celui d'exemples tests d'étiquette $transfert\_faible$ classés $transfert\_important$ et celui tests d'étiquette $transfert\_faible$ classés $transfert\_faible$. Nous avons aussi évalué les théories qui sont les unions de deux ou trois des théories $\mathcal{MOT}$, $\mathcal{AGR}$ et $\mathcal{COMB}$.

Théories	**Temps** CPU	**Nombre de règles**	**Classification sur les exemples test**	**Matrice de confusion**	
$\mathcal{MOT}$	$\simeq$ 18 h	28	81%	390	24
				197	558
$\mathcal{AGR}$	$\simeq$ 1 min	33	88%	479	38
				108	544
$\mathcal{COMB}$	$\simeq$ 35 h	34	87%	463	27
				124	555
$\mathcal{MOT} \cup \mathcal{AGR}$	-	61	90%	516	43
				71	539
$\mathcal{MOT} \cup \mathcal{COMB}$	-	62	89%	492	38
				95	544
$\mathcal{AGR} \cup \mathcal{COMB}$	-	67	91%	517	39
				70	543
$\mathcal{MOT} \cup \mathcal{AGR} \cup \mathcal{COMB}$	-	95	91%	526	46
				61	536

TAB. 3.26 – Les résultats en classification des théories $\mathcal{MOT}$, $\mathcal{AGR}$ et $\mathcal{COMB}$ et des théories constituées de l'union de ces trois premières.

Tout d'abord, on remarque que les apprentissages des théories constituées des motifs d'arbres, avec ou sans attribut agrégats sont très coûteux en temps (théories $\mathcal{MOT}$ et $\mathcal{COMB}$). Ce type de difficulté est souvent rencontré lors de recherches de règles à l'aide de la PLI, surtout lorsque le nombre d'exemples est important. Une première mise en œuvre de la recherche de motifs d'arbres a été effectué à l'aide du système de PLI ICL (De Raedt & Van Laer, 1995). Ce système, tout comme FOIL (sous-section 1.2.6.1), adopte une stratégie d'apprentissage de théorie de type "séparer pour régner" et la recherche d'une clause est de type descendante ; il paraît donc adapté à la recherche de motifs d'arbre telle que nous l'avons exposée. Mais l'opérateur de spécialisation de clauses défini dans la sous-section 3.4.3 ne peut pas être mis en œuvre au sein de ce système ; en particulier, l'énumération des arbres non ordonnés n'est pas possible ce qui rend le système très redondant dans la génération des hypothèses. Les résultats n'étaient pas meilleurs pour des temps de calculs plus importants.

La mise en œuvre d'une recherche de théories en logique attribut-valeur est, et de loin, le meilleur compromis entre résultats en classification et temps de calcul (théorie AGR). Si sa mise en place a nécessité un travail expert important pour la définition des attributs agrégats, celui-ci a été intéressant pour la découverte de nouveaux fac-

teurs pertinents. Ainsi, la surface maximale d'un exutoire de parcelle maïs (attribut $agr\_surf\_max\_mais$) a été identifié, tardivement, comme un attribut pertinent lors du processus de définition des attributs. Enfin, l'interprétation des motifs requiert des connaissances sur la mise en œuvre de l'induction. En particulier, il est nécessaire de bien comprendre la relation de couverture (voir l'exemple 3.9).

3.8 Conclusion

Dans ce chapitre, un premier objectif que nous nous sommes fixés est la définition d'un problème d'induction de règles à partir de résultats de simulation du modèle de transfert de pesticides SACADEAU. Pour y répondre, l'approche adoptée a été de présenter des résultats de simulation afin de motiver des problématiques d'apprentissage (chapitre 2). Une première problématique dégagée est l'influence des données climatiques, auxquelles nous nous sommes particulièrement intéressés lors d'un premier apprentissage (Cordier *et al.*, 2005a,b; Cordier, 2005). Une deuxième problématique a été dégagée dans le chapitre 2, il s'agit de l'influence des pratiques agricoles et de la localisation des parcelles cultivées en maïs.

Ce deuxième apprentissage est plus largement développé dans ce chapitre, nous avons en premier lieu constitué une base d'apprentissage composée d'arbres d'exutoires étiquetés à l'aide d'un taux de transfert moyenné sur les chroniques climatiques. Afin de caractériser une classe d'exemples, plusieurs approches sont adoptées (Trépos *et al.*, 2007). La première consiste à induire des motifs d'arbres d'exutoires, l'objectif étant de faire ressortir des relations d'écoulements pertinents entre exutoires. Une représentation des données d'apprentissage (exemples et règles) à l'aide d'une logique relationnelle telle que la logique des prédicats est nécessaire. Nous avons mis en œuvre une recherche descendante au sein du système de Programmation Logique Inductive pour la découverte des motifs. La deuxième approche repose sur la définition d'attributs agrégats pour synthétiser l'information contenue dans un arbre d'exutoires. Une dernière approche consiste à combiner les deux premières.

La deuxième approche semble un peu plus pertinente, d'une part parce que les règles induites sont plus faciles à interpréter, d'autre part, parce que la définition des attributs agrégats peut également être une étape de recherche des facteurs pertinents pour le transfert. Mais les deux premières approches sont complémentaires. L'approche PLI permet de se focaliser sur des connexions importantes entre parcelles, on voit apparaître essentiellement les données spatiales et topologiques. La deuxième permet de faire ressortir plus clairement les données de décision des agriculteurs : ces dernières semblent pertinentes à une échelle plus importante que la parcelle.

Chapitre 4

Analyses des règles pour l'aide à la décision

L'induction de règles est un sous-domaine important de l'apprentissage automatique. Malgré le fait qu'une règle soit exprimée dans un langage souvent plus facile à interpréter que ceux d'autres disciplines de l'apprentissage, l'analyse d'une théorie[1] de taille importante reste difficile. Lorsque les règles induites sont nombreuses et redondantes, il est intéressant de proposer des outils pour faciliter leur interprétation et aider un utilisateur dans sa prise de décisions. Nous nous focalisons ici sur l'analyse automatique ou interactive de théories induites, orientée aide à la prise de décisions. Des problématiques de la littérature liées à une telle analyse sont exposées dans la section 4.1.

Dans la section 4.2, nous proposons des méthodes d'extraction de recommandation d'actions. L'objectif est de laisser l'utilisateur proposer une situation défavorable (par exemple étiquetée $\ominus$), nous nous appuyons alors sur l'ensemble des règles de classification afin d'extraire des actions pour améliorer la situation. Les actions doivent, dans le cas idéal, permettre d'étiqueter la situation après action par une classe favorable $\oplus$. La faisabilité des actions suggérées doit être également prise en compte. Dans le cadre du développement de ces méthodes, nous nous restreignons à l'analyse de règles en logique attribut-valeur.

Enfin, nous proposons une interface de visualisation (section 4.3). On s'intéresse particulièrement à la relation de couverture entre les règles et les exemples. Concrètement, l'utilisateur peut souhaiter connaître les exemples couverts ou expliqués par une règle donnée ; ou inversement, connaître les règles qui pourraient constituer des explications de la classe d'un exemple donné. On s'appuie sur un langage de requête afin que l'utilisateur puisse spécifier ce qu'il veut voir apparaître. L'outil de recommandation d'action est également intégré dans cette interface de visualisation.

[1] une théorie est un ensemble de règles.

4.1 Analyse de règles pour l'aide à la décision

Un ensemble de règles de classification est souvent utilisé dans le cadre de la prédiction. La prédiction est la tâche qui consiste à assigner une étiquette à une situation. Mais l'utilisateur a aussi besoin d'une aide dans l'analyse des règles induites ; en particulier lorsque celles-ci sont nombreuses. Si les heuristiques utilisées lors de l'induction des règles permettent de comparer les règles entre elles[2], il existe d'autres mesures, telles que des mesures de similarité entre règles, sur lesquelles sont basés des outils de visualisation de règles (sous-section 4.1.1). La sélection de règles intéressantes est un moyen d'analyser des règles dans un objectif d'aide à la décision. On note toutefois que cette sélection peut être directement intégrée dans le processus d'induction ; il ne s'agit donc pas réellement d'une analyse de règles (sous-section 4.1.2). Dans la sous-section 4.1.3, les méthodes présentées cherchent à proposer des actions à partir des règles induites.

4.1.1 Visualisation des règles induites

La définition d'indices de similarité entre règles est un moyen de les comparer entre elles. Ces indices peuvent reposer sur leur syntaxe, c'est-à-dire, dans le cadre d'une logique attribut-valeur, sur les sélecteurs des complexes présents dans les règles. Ils peuvent aussi reposer sur la sémantique des règles, c'est-à-dire sur les ensembles d'exemples couverts par ces règles. Les indices de similarités reposant sur la sémantique des règles sont aussi utilisables dans le cas de règles en logique du premier ordre. Des visualisations s'appuyant sur ces similarités ont été développées. Par exemple, Gupta et al. (Gupta *et al.*, 1999) utilisent une technique de clustering hiérarchique pour organiser les règles et les visualiser. D'autres exemples sont l'utilisation de méthodes SOM (acronyme de l'anglais *Self Organizing Map*) ou MDS (acronyme de l'anglais *Multi-Dimensional Scaling*) afin de transposer les règles dans un plan 2D (un point représentant une règle) de telle manière que les distances entre les règles soient respectées au mieux (Tsumoto & Hirano, 2003; Rehm *et al.*, 2006; Gabriel *et al.*, 2006). Dans le but d'aider l'utilisateur à la prise de décision, une visualisation interactive des règles peut être intéressante. Zhao et al. (Zhao *et al.*, 2005) proposent une visualisation des règles et des données, dans une logique attribut-valeur, sous forme de matrice. Les lignes y représentent les étiquettes d'apprentissage et les colonnes les attributs. En repositionnant les cellules correspondant à l'intersection d'une classe d'apprentissage d'intérêt et d'un attribut sur lequel il peut agir, l'utilisateur peut se focaliser sur les règles dans un objectif de prise de décision. Une autre visualisation proche de celle-ci est proposée dans (Han & Cercone, 2000).

4.1.2 Sélection de règles intéressantes

Une notion importante dans le cadre de la recherche de règles intéressantes est celle d'*actionnabilite* proposée par Silberschatz et Tuzhilin (Silberschatz & Tuzhilin, 1996).

[2] Une analyse des heuristiques, telles que le $m-estimate$ (exemple 1.19), est proposée par Fürnkranz et Flach (Fürnkranz & Flach, 2003).

Étant donné un motif, par exemple une règle, il s'agit de savoir s'il est pertinent vis-à-vis d'une action à entreprendre. D'après eux, un motif est intéressant, d'un point de vue subjectif, s'il est soit :

- manipulable : l'utilisateur peut utiliser le motif afin d'agir dans son intérêt ;
- inattendu : l'utilisateur est surpris par le motif induit.

Si de nombreux travaux se sont intéressés à la recherche de règles inattendues, voir par exemple (Duval *et al.*, 2007), la notion de motif "manipulable" est moins abordée. Pour la découverte de règles intéressantes, Chen et Liu (Chen & Liu, 2001) suggèrent d'intégrer des connaissances de l'utilisateur a priori. Simplement, dans une étape de pré-apprentissage, les règles attendues, données par l'utilisateur, permettent de cibler les exemples qui ne sont pas conformes aux connaissances a priori. De cette manière, les règles inattendues issues d'un apprentissage classique sont repérées : ce sont celles qui couvrent des exemples non conformes. Des algorithmes d'apprentissage de règles ont été développés pour le cas où l'on s'intéresse à un sous-groupe de données. Par exemple, Lavrač et al. (Lavrac *et al.*, 2002) adaptent le système CN2 (voir sous-section 1.1.2) pour la découverte de sous-groupes (de l'anglais *subgroup discovery*). Des poids importants sont associés aux exemples du sous-groupe ciblé, ce qui implique quelques modifications lors du calcul de l'heuristique de recherche. Au contraire, étant donné un ensemble de groupes de données $\{G_1, .., G_n\}$, Bay et Pazzani (Bay & Pazzani, 2001) induisent des complexes contrastant d'un groupe à l'autre (en anglais *contrast set*) ; c'est à dire que le support de la règle restreint à un groupe de données G_i doit être suffisamment plus important que son support restreint à un groupe G_j $(i \neq j)$.

4.1.3 Proposition d'actions

Des travaux se focalisent sur l'extraction d'actions à partir des règles induites. L'idée générale est, partant d'une situation défavorable, de proposer des modifications à apporter pour améliorer la classe de la situation. On se place ici dans le cadre de règles en logique attribut-valeur et une situation est représentée par un complexe (voir section 1.1).

Yang et al. (Yang *et al.*, 2003) suggèrent d'utiliser des arbres de décision pour construire des actions. Plus précisément, étant donné un exemple (situation) couvert par une règle de classe c_1, des modifications dans les valeurs d'attribut sont proposées afin que l'exemple soit couvert par une règle de classe c_2 considérée comme meilleure. Une mesure de profit P_N est proposée : $P_N = P_E * P_{gain} - \sum Cost$. P_E est le profit engrangé lorsque l'exemple passe de la classe c_1 à c_2 ; P_{gain} est la probabilité que l'individu passe effectivement de la classe c_1 à c_2 ; $\sum Cost$ est la somme des coûts des modifications unitaires. Une matrice, fournissant pour chaque attribut le coût de la modification d'une valeur de l'attribut v_i en une valeur v_j, doit en effet être définie. Cette approche contraint à l'utilisation d'attributs qualitatifs uniquement. Une autre approche (Ras & Wieczorkowska, 2000) consiste à induire des règles actions. Pour tout couple de règles induites $r_1 = comp_1 \rightarrow c_1$ et $r_2 = comp_2 \rightarrow c_2$, une règle d'action est définie, dans le cas où la classe c_2 est considérée comme meilleure que c_1 ; cette règle

s'écrit[3] $(comp_1 => comp_2) \rightarrow (c_1 => c_2)$ et s'interprète de la manière suivante : "si l'on modifie les valeurs des sélecteurs du complexe $comp_1$ de telle manière à être couvert par le complexe $comp_2$ alors la classe passe de c_1 à c_2". Les auteurs identifient deux types d'attributs, les attributs *flexibles* et *stables*. Si la règle action nécessite la modification d'un attribut stable, alors celle-ci n'est pas proposée à l'utilisateur car on la considère infaisable. Une extension (Ras & Tsay, 2003) propose de telles actions lorsque la modification de la valeur de l'attribut stable est en fait une spécialisation ; dans ce cas, le sélecteur sur cet attribut peut être interprété comme un contexte dans lequel l'individu, sur lequel on applique la règle action, doit se trouver.

4.2 Recommandation d'actions en logique attribut-valeur

Dans cette section, on s'appuie sur les travaux de la littérature pour définir une tâche originale de recommandation d'actions. Nous adoptons en partie le point de vue de Yang et al. (Yang *et al.*, 2003). En effet, nous considérons que l'utilisateur propose une situation qui lui est défavorable, en général étiquetée par la classe $\ominus$. Pour proposer des actions, notre démarche ne repose pas sur les arbres de décision mais plus généralement sur un ensemble de règles de classification ; de ce point de vue, elle est plus proche de celle proposée dans (Ras & Wieczorkowska, 2000). De même que pour ces deux approches, nous nous plaçons dans une logique attribut-valeur et considérons que les actions sont des modifications de valeur des attributs décrivant la situation défavorable. Pour évaluer les actions, Yang et al. (Yang *et al.*, 2003) définissent une fonction de profit. De notre côté, nous préférons utiliser les deux notions de *qualité* et *faisabilité* d'une action.

Étant donnés une situation et un ensemble de règles, un espace de recherche des actions et un critère pour les évaluer sont définis (sous-section 4.2.3). Nous nous intéressons particulièrement à la notion de faisabilité des actions dans la sous-section 4.2.2. Enfin deux algorithmes pour induire les actions sont proposées : le premier s'appuie sur une recherche d'actions par faisceau (sous-section 4.2.4), le second a pour objectif de construire l'action optimale (au sens du critère de qualité) dans l'espace de recherche. Avant de les présenter, il est nécessaire d'introduire quelques définitions.

4.2.1 Définitions préliminaires

Les symboles $\oplus$ et $\ominus$ dénotent les étiquettes de classes jugées comme satisfaisante et non satisfaisante. La théorie $\mathcal{R}$ préalablement induite est constituée de règles des deux classes : $\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus}$. L'ensemble des attributs est $\mathcal{X} = \{X_1, \ldots, X_n\}$ dont les domaines, notés $Dom_1, \ldots, Dom_n$, sont des valeurs qualitatives ou quantitatives.

Une *valeur qualitative* est un ensemble de constantes. Une *valeur quantitative* est un intervalle représenté par un quadruplet $< min, max, l, r >$ où min et max sont des réels représentant les bornes minimales et maximales de l'intervalle ; l et r sont des valeurs de $\{-1, 1\}$, l (respectivement r) étant à 1 lorsque la valeur min (respectivement max)

[3] Pour faciliter la lecture, la syntaxe des règles action est simplifiée.

fait partie de l'intervalle et -1 sinon. On utilise essentiellement une notation courante pour les intervalles ; par exemple, l'intervalle $< 2, 3, -1, 1 >$ pourra s'écrire $]2, 3]$. Ma et Hayes (Ma & Hayes, 2006) redéfinissent les treize relations d'Allen entre intervalles (Allen, 1981) pour le cas où l'information sur l'appartenance ou non d'une borne dans l'intervalle est explicite. La figure 4.1 présente ces relations de manière non exhaustive.

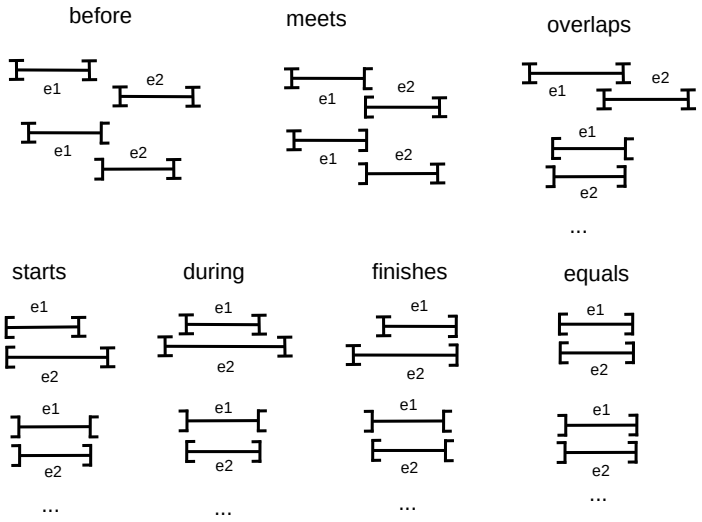


FIG. 4.1 – Exemples de couple d'intervalles e_1, e_2 et leur relation d'Allen redéfinie pour les intervalles dont les bornes peuvent être exclues (Ma & Hayes, 2006). Pour une borne donnée dont la valeur est v, le symbole **I** indique que v peut être incluse ou non dans l'intervalle ; le symbole] , lorsqu'il est à gauche, signifie que v est exclue etc... Les six autres relations peuvent être définies de la manière suivante : (e_1 after e_2) ssi (e_1 before e_2), (e_1 met-by e_2) ssi (e_1 meet e_2), (e_1 overlapped-by e_2) ssi (e_1 overlaps e_2), (e_1 started-by e_2) ssi (e_1 starts e_2), (e_1 contains e_2) ssi (e_1 during e_2), (e_1 finished-by e_2) ssi (e_1 finishes e_2).

La *taille* d'une valeur qualitative est le nombre de constantes de l'ensemble. La *taille* d'une valeur quantitative $< min, max, l, r >$ est $max - min$. On note $\emptyset$ la valeur vide, $\mathcal{V}_l$ (respectivement $\mathcal{V}_t$) l'ensemble (sans $\emptyset$) des valeurs qualitatives (respectivement quantitatives). Nous présentons maintenant les opérateurs d'intersection et d'union.

Définition 4.2 (Opérateurs $\boxtimes$ et $\boxplus$ entre valeurs).
Les opérateurs $\boxtimes$ et $\boxplus$ ont pour signature $\mathcal{V} \cup \{\emptyset\} * \mathcal{V} \cup \{\emptyset\} \rightarrow \mathcal{V} \cup \{\emptyset\}$ où $\mathcal{V}$ est fixé soit à $\mathcal{V}_t$, soit à $\mathcal{V}_l$. Ils sont respectivement les opérateurs usuels d'intersection et d'union, pour les ensembles dans le cas de valeurs qualitatives et pour les intervalles dans le cas de valeurs quantitatives.

La définition de ces opérateurs nous permet de manipuler les valeurs indépendemment de leur type. On peut par exemple définir l'inclusion entre valeurs qualitatives ou quantitatives.

Définition 4.3 (Inclusion et inclusion stricte entre valeurs).
Soient e_1 et e_2 deux valeurs quantitatives ou qualitatives, e_1 est incluse dans e_2 (on note $e_1 \subseteq e_2$) ssi $e_1 \boxtimes e_2 = e_1$; e_1 est incluse strictement dans e_2 (on note $e_1 \subset e_2$) si $e_1 \subseteq e_2$ et $e_1 \neq e_2$.

L'opérateur de différence $\boxminus$ que nous utilisons est un opérateur plus spécifique à nos besoins.

Définition 4.4 (Opérateurs $\boxminus$ entre valeurs).
L'opérateur $\boxminus$ a pour signature $\mathcal{V} * \mathcal{V} \rightarrow 2^{\mathcal{V}}$ où $\mathcal{V}$ est fixé soit à $\mathcal{V}_t$, soit à $\mathcal{V}_l$. Soient $e_1, e_2 \in \mathcal{V}$, $e_1 \boxminus e_2$ est l'ensemble[4] des valeurs v telles que :
- *$\forall v \in e_1 \boxminus e_2$, $v \boxtimes e_1 = v$ (c'est-à-dire $v \subseteq e_1$) et $v \boxtimes e_2 = \emptyset$;*
- *$\forall v'$ tel que $v' \boxtimes e_1 = v'$ et $v' \boxtimes e_2 = \emptyset$, $\exists e \in e_1 \boxminus e_2$ tel que $v' \subseteq e$.*

Si e_1 et e_2 sont qualitatifs on note $d = (e_1 - e_2)$ la différence ensembliste usuelle alors $e_1 \boxminus e_2$ est égal à $\{d\}$ si d n'est pas vide et $\{\}$ sinon. Pour les valeurs qualitatives le résultat de $\boxminus$ est donc un ensemble de valeurs de taille 0 ou 1. Pour les valeurs quantitatives, cet ensemble peut être de taille 0, 1 ou 2 : il est calculé en fonction de la relation entre e_1 et e_2 (tableau 4.5).

Relations entre e_1 et e_2	$e_1 \boxplus e_2$	$e_1 \boxtimes e_2$	$e_1 \boxminus e_2$
e_1 before e_2 e_1 meets e_2	$\langle mi_1, ma_2, l_1, r_2\rangle$	$\emptyset$	$\{\langle mi_1, ma_1, l_1, r_1\rangle\}$
e_1 overlaps e_2 e_1 finished-by e_2	$\langle mi_1, ma_2, l_1, r_2\rangle$	$\langle mi_2, ma_1, l_2, r_1\rangle$	$\{\langle mi_1, mi_2, l_1, -l_2\rangle\}$
e_1 starts e_2 e_1 during e_2 e_1 finishes e_2 e_1 equals e_2 ($e_1 = e_2$)	$\langle mi_2, ma_2, l_2, r_2\rangle$	$\langle mi_1, ma_1, l_1, r_1\rangle$	$\{\}$
e_1 after e_2 e_1 met-by e_2	$\langle mi_2, ma_1, l_2, r_1\rangle$	$\emptyset$	$\{\langle mi_1, ma_1, l_1, r_1\rangle\}$
e_1 overlapped-by e_2 e_1 started-by e_2	$\langle mi_2, ma_1, l_2, r_1\rangle$	$\langle mi_1, ma_2, l_1, r_2\rangle$	$\{\langle ma_2, ma_1, -r_2, r_1\rangle\}$
e_1 contains e_2	$\langle mi_1, ma_1, l_1, r_1\rangle$	$\langle mi_2, ma_2, l_2, r_2\rangle$	$\{\langle mi_1, mi_2, l_1, -l_2\rangle,$ $\langle ma_2, ma_1, -r_2, r_1\rangle\}$

TAB. 4.5 – Les opérateurs $\boxplus$, $\boxtimes$ et $\boxminus$ pour les valeurs quantitatives $e_1 = \langle mi_1, ma_1, l_1, r_1\rangle$ et $e_2 = \langle mi_2, ma_2, l_2, r_2\rangle$. On a également, pour une valeur e (qualitative ou quantitative) : $e \boxtimes \emptyset = \emptyset \boxtimes e = \emptyset$; $e \boxplus \emptyset = \emptyset \boxplus e = e$.

[4]Le résultat de $\boxminus$ est un ensemble de valeurs. Ainsi, lorsque e_1 et e_2 sont des valeurs qualitatives, le résultat de $\boxminus$ est un ensemble d'ensemble de constantes. Si elles sont des valeurs quantitatives, le résultat est un ensemble d'intervalles.

Les opérateurs $\boxtimes$ et $\boxplus$ sont les opérateurs d'union et d'intersection bien connus ; en particulier, ils sont associatifs. Au contraire l'opérateur $\boxminus$, qui n'est pas associatif, est spécifique à notre approche (voir l'exemple 4.7). La propriété suivante sur $\boxminus$ est nécessaire pour la suite.

Proposition 4.6:
Considérons deux valeurs e_1 et e_2 de même type (qualitatif ou quantitatif). Si $e_1 \not\subseteq e_2$ et $e_2 \not\subset e_1$ alors $e_2 \boxminus e_1$ contient exactement 1 valeur.

PREUVE. On a donc soit :

- e_1 et e_2 sont des ensembles de constantes, comme $e_2 \not\subset e_1$ alors il existe au moins une constante de e_2 non présente dans e_1. Par définition, on a $e_1 - e_2$ est non vide et $e_1 \boxminus e_2$ est de taille 1.
- e_1 et e_2 sont des intervalles, on cherche alors la relation entre e_1 et e_2. Comme $e_1 \not\subseteq e_2$ on a $e_1 \boxtimes e_2 \neq e_1$, et, à l'aide du tableau 4.5, on déduit que cette relation ne peut pas être *starts*,*during*, *finishes* ou *equals*. Si la relation était *contains* alors $e_1 \boxtimes e_2 = e_2 \boxtimes e_1 = e_2$ et $e_2 \subseteq e_1$. On a vu que la relation ne pouvait pas être *equals* alors on a $e_2 \subset e_1$ ce qui est contradictoire avec l'hypothèse. Pour toutes les autres relations possibles l'ensemble $e_2 \boxminus e_1$ ne contient qu'un intervalle.

Exemple 4.7 (relations entre valeurs quantitatives, opérateurs $\boxtimes$, $\boxplus$ et $\boxminus$).
Considérons les trois intervalles $e_{t1} = [2, 5]$, $e_{t2} =]3, 6]$ et $e_{t3} = [4, 6[$. La relation entre e_{t1} et e_{t2}, ainsi qu'entre e_{t1} et e_{t3}, est *overlaps*, celle entre e_{t2} et e_{t3} est *contains*. D'autres relations entre intervalles sont illustrées dans la figure 4.1. Pour les opérateurs classiques d'intersection et d'union, on a $e_{t1} \boxtimes e_{t2} =]3, 5]$, $e_{t1} \boxplus e_{t2} = [2, 6]$. Pour l'opérateur de différence, on a $e_{t1} \boxminus e_{t2} = \{[2, 3]\}$, $e_{t2} \boxminus e_{t3} = \{]3, 4[, [6, 6]\}$ et $e_{t3} \boxminus e_{t2} = \{\}$. Considérons maintenant les deux valeurs qualitatives $e_{l1} = \{a, b\}$ et $e_{l2} = \{a\}$ alors $e_{l1} \boxtimes e_{l2} = \{a\}$, $e_{l1} \boxplus e_{l2} = \{a, b\}$ et $e_{l1} \boxminus e_{l2} = \{\{b\}\}$.

Nous redéfinissons à présent les *sélecteurs* et *complexes* comme des généralisations de ceux introduits dans la section 1.1. Le formalisme présenté ici nous permet en effet de les exprimer.

Définition 4.8 (Sélecteur et complexe).
Pour un attribut $X_i \in \mathcal{X}$, un sélecteur est noté $(X_i \in d_i)$ où $d_i \subseteq Dom_i$. Un complexe est noté $C = (X_{k_1} \in d_{k_1}) \wedge \ldots \wedge (X_{k_m} \in d_{k_m})$ où pour tout i, $k_i \in \{1, \ldots, n\}$ et $d_{k_i} \subseteq Dom_{k_i}$.

Un complexe peut avoir plusieurs sélecteurs pour un même attribut, c'est pourquoi nous introduisons la notion de projection sur un attribut et redéfinissons la notion de couverture.

Définition 4.9 (Projection sur attribut et complexe cohérent).
Soient C un complexe et X_i un attribut, l'ensemble des sélecteurs (possiblement vide)

de C sur l'attribut X_i est noté $sel_i = \{X_i \in d_{i_1}, \ldots, X_i \in d_{i_m}\}$. La projection de C sur X_i, notée $C[X_i]$, est définie par :

$$C[X_i] = \begin{cases} Dom_i & si\ sel_i = \{\} \\ d_{i_1} \boxtimes \ldots \boxtimes d_{i_m} & sinon \end{cases}$$

Exemple 4.10 (sélecteurs, complexes et projections).
Soient X_1 et X_3 deux attributs qualitatifs de domaines respectivement $Dom_1 = \{a, b, c, d\}$ et $Dom_3 = \{p, q\}$. Soit X_2 un attribut quantitatif de domaine $Dom_2 = [0, 10]$.
On considère le complexe suivant constitué de trois sélecteurs :
$C = (X_1 \in \{a, b\}) \wedge (X_2 \in [0, 3]) \wedge (X_2 \in [1, 5])$
Alors $sel_2 = \{(X_2 \in [0, 3]), (X_2 \in [1, 5])\}$, les projections sur les attributs sont :
$C[X_1] = \{a, b\}$, $C[X_2] = [1, 3]$ et $C[X_3] = \{p, q\}$

Au sein d'un complexe, il peut exister une contradiction entre deux sélecteurs basés sur un même attribut. C'est pourquoi nous introduisons la définition de complexe *cohérent*.

Définition 4.11 (Complexe cohérent).
Un complexe C est cohérent si : $\forall i, C[X_i] \neq \emptyset$.

La relation de couverture doit être redéfinie : il ne s'agit plus en effet d'une simple inclusion des sélecteurs d'un complexe dans un autre.

Définition 4.12 (Couverture et équivalence).
Soient C et C' deux complexes, C couvre C' si et seulement si : $\forall i, C'[X_i] \subseteq C[X_i]$. C et C' sont équivalents si et seulement si C couvre C' et C' couvre C.

On note trivialement, à l'instar de la relation de couverture définie dans la section 1.1, que si les sélecteurs d'un complexe C sont présents dans un complexe C' alors C couvre C'. De plus, si on définit le complexe P par $P = C[X_1] \wedge \ldots \wedge C[X_n]$, alors C et P sont équivalents.

Pour simplifier l'écriture, on étend l'opérateur $\boxtimes$ défini pour les valeurs aux complexes. On définit également une relation de *connexion* entre complexes utile dans la section 4.2.5. Intuitivement, deux complexes sont connectés s'ils s'intersectent.

Définition 4.13 (Opérateur $\overline{\boxtimes}$ et complexes connectés).
Soit C et C' deux complexes, le résultat de l'opérateur $\overline{\boxtimes}$ entre C et C' est :
$C \overline{\boxtimes} C' = X_1 \in (C[X_1] \boxtimes C'[X_1]) \wedge \ldots \wedge X_n \in (C[X_n] \boxtimes C'[X_n])$.
C et C' sont connectés si $C \overline{\boxtimes} C'$ est cohérent.

Des sélecteurs, complexes, et les relations entre complexes sont exposés ici pour l'exemple.

Exemple 4.14 (couverture et connexion).
Considérons les attributs X_1 et X_2 de l'exemple 4.10. Soient les deux complexes :
$C = (X_1 \in \{a, b\}) \wedge (X_1 \in \{a\}) \wedge (X_2 \in [2, 5])$

$C' = (X_1 \in \{a\}) \wedge (X_2 \in [3,6[)$. Alors, on a :
$C[X_1] = \{a\}$, $C[X_2] = [2,5]$, $C'[X_1] = \{a\}$, $C'[X_2] = [3,6[$
C n'est pas couvert par C' puisque $C[X_2] \not\subseteq C'[X_2]$. C et C' sont connectés puisque $C\overline{\boxtimes}C'[X_1] = \{a\} \neq \emptyset$ et $C\overline{\boxtimes}C'[X_2] = X_2 \in [3,5] \neq \emptyset$.

Nous pouvons à présent clarifier le vocabulaire utilisé par la suite, en particulier ce que l'on appelle situation, action et règle de classification.

Définition 4.15 (Situation, action et règle de classification).
Une situation est un complexe. Une action est un complexe. Une règle de classification R est un complexe, mais on parle également de la classe de R qui peut être $\oplus$ ou $\ominus$.

Une action, appliquée à une situation, produit un complexe que l'on appelle *situation résultante.*

Définition 4.16 (Situation résultante, correction et précision).
Soit S une situation et A une action, le résultat de l'application de A sur S est un complexe noté $res(S,A) = X_1 \in r_1 \wedge \ldots \wedge X_n \in r_n$ où :

$$r_i = \begin{cases} A[X_i] & \text{si } S[X_i] \boxtimes A[X_i] = \emptyset \\ S[X_i] \boxtimes A[X_i] & \text{sinon} \end{cases}$$

Dans le premier cas, lorsque $S[X_i] \boxtimes A[X_i] = \emptyset$, on dit que l'action A est une correction de S sur l'attribut X_i. Dans le deuxième cas, A consiste en une précision de S sur l'attribut X_i ; on a en effet $res(S,A)[X_i] \subseteq S[X_i]$.

Exemple 4.17 (situation résultante).
Considérons les attributs X_1 et X_2 de l'exemple 4.10. Soit la situation S :
$S = (X_1 \in \{a\}) \wedge (X_2 \in [2,5])$.
Le résultat de l'action $A = (X_1 \in \{b\} \wedge X_2 \in]2,8])$ sur S est :
$res(S,A) = (X_1 \in \{b\}) \wedge (X_2 \in]2,5])$
A est une correction sur l'attribut X_1 et c'est une précision sur l'attribut X_2
Le résultat de l'action $A' = (X_2 \in [6,7])$ sur S est :
$res(S,A') = (X_1 \in \{a\}) \wedge (X_2 \in [6,7])$.
A' est une correction sur l'attribut X_2

4.2.2 Faisabilité d'une action

Nous nous intéressons ici à la notion de faisabilité d'une action. Dans l'approche proposée dans (Ras & Wieczorkowska, 2000; Ras & Tsay, 2003), un concept de *flexibilité* est utilisé pour découvrir des règles d'actions. Deux types d'attributs sont distingués : les attributs *stables* sont les attributs qui ne peuvent être modifiés (e.g. sexe) contrairement aux attributs *flexibles* (e.g. valeur d'un prêt bancaire). Une règle d'action dans ce cadre porte naturellement sur des attributs flexibles afin de permettre à l'utilisateur d'agir. L'action est donc faisable si elle propose des modifications pour les attributs flexibles uniquement.

Définition 4.18 (Faisabilité par restriction).
Soit $FLEX$ le sous-ensemble des attributs flexibles, une action A est dite faisable par restriction sur $FLEX$ si $\forall X_i \in (\mathcal{X} - FLEX), A[X_i] = Dom_i$. L'action ne doit pas consister en une modification d'un attribut stable. L'ensemble $FLEX$ est un paramètre utilisateur.

Cette nette distinction entre attributs flexibles et stables est certes intéressante, mais à notre sens, assez rigide et pas toujours suffisante. Par exemple, un médecin préférerait prescrire des médicaments à un patient malade plutôt que d'envisager pour lui une intervention chirurgicale si celle-ci peut être évitée. L'action de prescrire des médicament paraît *plus* faisable. Une approche possible est de tenir compte de cette différence entre les attributs en affectant à chaque attribut des *poids de stabilité* w_i. Ce poids mesure, en quelque sorte, à quel point on ne peut pas agir sur cette propriété. D'un autre côté, un médecin souhaitant prescrire un régime alimentaire à un patient, prend en considération le poids que le patient doit perdre. S'il s'agit de faire perdre au patient peu de poids, e.g. 2 kg, alors un régime est tout à fait faisable. En revanche, une perte de 80 kg est un but très difficile à atteindre par un simple régime. C'est pourquoi une distance nous semble tout à fait adaptée à notre problème. Ce dernier, rappelons le, consiste à proposer des actions réalisables à accomplir face à une situation jugée non satisfaisante, et qui conduirait à une situation résultante "assez proche" de la situation initiale mais plus satisfaisante.

La littérature regorge de définitions de distances entre complexes, par exemple (De Carvalho, 1994; Gowda & Diday, 1991; Ichino & Yaguchi, 1994). En nous basant sur une étude empirique réalisée par Malerba et al. (Malerba *et al.*, 2001), notre choix s'est porté sur la distance généralisée de Minkowski proposée par Ichino et Yaguchi dans (Ichino & Yaguchi, 1994). Cette mesure convient à notre problème car, non seulement elle gère les valeurs qualitatives et quantitatives, mais de plus, elle permet d'intégrer des poids de stabilité dans le calcul de la dissimilarité entre complexes. Les auteurs proposent une mesure de dissimilarité entre deux valeurs c_i et c'_i que nous présentons ici à l'aide des opérateurs définis ci-dessus :

$$\phi(c_i, c'_i) = |c_i \boxplus c'_i| - |c_i \boxtimes c'_i| + \gamma(2|c_i \boxtimes c'_i| - |c_i| - |c'_i|)$$

où $|c_i|$ dénote la taille de c_i et $\gamma \in [0, 0.5]$ contrôle la proximité interne et externe entre c_i et c'_i. Pour deux complexes C et C', Ichino et Yagushi définissent la mesure de dissimilarité pondérée et normalisée appelée distance généralisée de Minkowski d'ordre p comme suit :

$$dist_p(C, C') = \left[\sum_{i=1}^{n} \{w_i \ \frac{\phi(C[X_i], C'[X_i])}{|Dom_i|}\}^p\right]^{1/p}$$

où les poids de flexibilité $w_i > 0$, $i \in \{1, .., n\}$ sont choisis tels que $\sum_{i=1}^{n} w_i = 1$ et où la distance généralisée de Minkowski satisfait $0 \leq dist_p(C, C') \leq 1$. Enfin, il est prouvé que $dist_p(C, C')$ satisfait tous les axiomes d'une métrique. Nous proposons ici une deuxième définition de la faisabilité basée sur cette distance entre complexes[5].

[5]Nous fixons $p = 2$ et $\gamma = 0.25$ (Malerba *et al.*, 2001).

Définition 4.19 (δ-faisabilité).
Une action A est dite δ-faisable relativement à une situation S (où δ est un paramètre réel entre 0 et 1) si $dist_p(S, res(S, A)) < \delta$. Les poids w_i et le seuil δ sont des paramètres utilisateurs.

Pour la suite, nous définissons une propriété de monotonie pour les définitions de faisabilité. Cette propriété est utilisée dans la sous-section 4.2.5.

Définition 4.20 (Monotonie de la faisabilité).
La faisabilité est monotone si pour tout couple d'actions A et A' tel que A couvre A', A n'est pas faisable implique que A' n'est pas faisable[6].

Proposition 4.21:
La faisabilité par restriction est monotone et la δ-faisabilité n'est pas monotone.

PREUVE. Monotonie de la faisabilité par restriction.
Si A n'est pas faisable alors $\exists X_i \in \mathcal{X} - FLEX$ tel que $A[X_i]$ est différent de Dom_i donc par définition $A[X_i] \subset Dom_i$. Comme $A'[X_i] \subseteq A[X_i]$ alors $A'[X_i] \subset Dom_i$ et A' n'est pas faisable.
Non monotonie de la δ-faisabilité.
On montre pour cela un contre exemple. Posons $\mathcal{X} = \{X_1\}$ où X_1 est un attribut quantitatif, $Dom_1 = [2, 6]$, $w_1 = 1$, $\delta = 12/16$ et une situation $S = (X_1 \in [2, 3])$. Considérons l'action $A = (X \in [4, 6])$ alors $dist_2(S, res(S, A)) = 13/16 > \delta$ et A n'est pas faisable. Soit maintenant l'action $A' = (X \in [4, 5])$ on a $dist_2(S, res(S, A')) = 10/16 < \delta$ et A' est faisable. Si la δ-faisabilité était monotone, comme A n'est pas faisable et que $[4, 5] \subseteq [4, 6]$, A' ne devrait pas être faisable.

4.2.3 Tâche de recherche d'actions

Étant donné une théorie $\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus}$ et une situation S non satisfaisante, en général S est classée dans $\ominus$ par la théorie, l'objectif est de proposer des actions à appliquer à S telle que la situation résultante soit classée dans $\oplus$.

Pour formaliser la tâche d'apprentissage, l'espace de recherche des actions doit être défini et, de plus, un critère de qualité des actions est nécessaire pour comparer les actions entre elles. L'espace de recherche que nous considérons est l'ensemble des actions *valides*.

Définition 4.22 (Action valide).
Soit S et A deux complexes, l'action A est valide relativement à S si : $\forall i, (A[X_i] \neq Dom_i) \rightarrow (A[X_i] \boxtimes S[X_i] = \emptyset)$.

Si l'action A propose une modification de la valeur d'un attribut, cette modification doit être une correction et non une précision pour que A soit valide (voir définition 4.16). Cela correspond à un choix que nous avons fait. Lorsque la situation S est totalement

[6] On a donc également : A' est faisable implique que A est faisable.

instanciée[7], il n'existe de toute façon pas d'actions constituées de précisions ; ce qui est le cas que nous traitons. Dans l'exemple 4.17, l'action A' est valide mais pas l'action A.

Pour évaluer une action A, on estime en fait la différence entre la confiance de la classification de $res(S, A)$ dans la classe $\oplus$ et la confiance de sa classification dans $\ominus$.

Définition 4.23 (Qualité d'une action).
La qualité d'une action A pour une situation S repose sur la confiance des règles qui couvrent la situation résultante $res(S, A)$:

$$qualite(S, A) = \sum_{\substack{R \in \mathcal{R}^{\oplus}, \\ R \text{ couvre } res(S,A)}} conf(R) \; - \sum_{\substack{R \in \mathcal{R}^{\ominus}, \\ R \text{ couvre } res(S,A)}} conf(R)$$

Les définitions précédentes sont suffisantes pour formaliser la tâche sur laquelle nous nous focalisons dans cette section.

Définition 4.24 (Tâche de recommandation d'actions).
Étant donné une théorie $\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus}$ et une situation S non satisfaisante, en général de classe $\ominus$, la recherche d'actions consiste à construire un ensemble de M actions cohérentes, valides et faisables. Les actions sont évaluées et comparées à l'aide du critère de qualité. Nous souhaitons qu'au moins une action qui maximise le critère de qualité dans l'espace des actions cohérentes, valides et faisables soit proposée parmi les M actions. La taille M de l'ensemble des actions proposées est un paramètre utilisateur.

L'espace de recherche des actions est constitué de l'ensemble des actions cohérentes et valides. Dans cet espace, il s'agit alors de proposer à l'utilisateur des actions faisables.

4.2.4 Un premier algorithme pour la recherche d'actions

Dans cette première approche, nous constituons un ensemble d'actions élémentaires $\mathcal{A}$, c'est-à-dire des actions possédant un seul sélecteur. L'idée de la recherche est alors de combiner ces actions élémentaires entre elles afin de former des actions plus complexes.

Afin de proposer uniquement des actions valides et cohérentes, les actions élémentaires doivent elles même être valides et cohérentes. On s'appuie pour les définir sur les sélecteurs présents dans les règles de $\mathcal{R}^{\oplus}$ et l'opérateur de différence $\boxminus$ (voir définition 4.4).

Définition 4.25 (Actions élémentaires $\mathcal{A}$).
Soit une situation S et un ensemble de règles $\mathcal{R}^{\oplus}$, l'ensemble des actions élémentaires est défini comme :

$$\mathcal{A} = \bigcup_{R \in \mathcal{R}^{\oplus}} discr(R, S)$$

[7] Un complexe est totalement instancié lorsque les valeurs des projections contiennent uniquement un point pour le cas quantitatif et une constante pour le cas qualitatif. Ainsi $(X_1 \in \{a\}) \wedge (X_2 \in [5, 5])$ est totalement instancié.

Où $discr(S, R)$ est un ensemble discriminant maximal de sélecteurs pour R et S[8] *:*

$$discr(R, S) = \bigcup_{i=1}^{i=n} \bigcup_{\Delta \in R[X_i] \boxminus S[X_i]} \{(X_i \in \Delta)\}$$

Pour tout sous-ensemble $E = \{sel_1, \ldots, sel_e\}$ inclus dans $\mathcal{A}$, il est possible de construire une action $(sel_1 \wedge \ldots \wedge sel_e)$ qui est valide par construction mais pas nécessairement cohérente.

Exemple 4.26 (actions élémentaires).
Soient un attribut qualitatif X_l de domaine $\{a, b, c\}$, un attribut quantitatif X_t de domaine $[0, 10]$, une situation $S = (X_l \in \{a\}) \wedge (X_t \in [5, 5])$ et deux règles de classe $\oplus$: $R_1^{\oplus} = (X_l \in \{a, b\}) \wedge (X_t \in [5, 7]) \rightarrow \oplus$ et $R_2^{\oplus} = (X_t \in [4, 6]) \rightarrow \oplus$. On a alors
$discr(R_1^{\oplus}, S) = \{X_l \in \{b\}, X_t \in]5, 7]\}$
$discr(R_2^{\oplus}, S) = \{X_l \in \{b, c\}, X_t \in [4, 5[, X_t \in]5, 6]\}$
$\mathcal{A} = \{X_l \in \{b\}, X_t \in]5, 7], X_l \in \{b, c\}, X_t \in [4, 5[, X_t \in]5, 6]\}$
On peut former l'action $(X_t \in [4, 5[) \wedge (X_t \in]5, 7])$ qui n'est en l'occurrence pas cohérente.

L'algorithme 4.27 élabore des actions constituées d'un sous-ensemble des sélecteurs de $\mathcal{A}$ et retient les actions cohérentes de meilleure qualité. Il explore l'espace des actions en utilisant une stratégie de recherche par *faisceau*. En fait, il maintient un ensemble (nommé faisceau) de taille M des actions de meilleure qualité que l'algorithme construit. Au départ le faisceau est initialisé à $\{vrai\}$ ($vrai$ étant l'action constituée d'aucun sélecteur). Afin d'explorer l'espace des actions, l'algorithme spécialise les actions du faisceau en s'appuyant sur l'ajout de sélecteurs de $\mathcal{A}$. Une action est retenue dans le faisceau si elle est cohérente et faisable ; l'action est de toute façon valide puisque les actions élémentaires sont valides. L'algorithme rejette du faisceau l'action de qualité la plus faible lorsque le faisceau dépasse une taille fixée par l'utilisateur ; en l'occurrence M, le nombre d'actions souhaitées.

4.2.5 Recherche de l'optimum

Dans l'approche précédente, les actions proposées à l'utilisateur peuvent être très ressemblantes. En effet, rien ne contraint l'algorithme à construire des actions pour lesquelles les situations résultantes soient couvertes par des ensembles de règles différents. Ainsi, les actions proposées peuvent être les mêmes à quelques légères différences, par exemple dans les bornes des intervalles présents dans les sélecteurs.

C'est pourquoi l'approche proposée ici s'appuie sur plusieurs règles pour la construction d'une seule action dite *composite*.

Définition 4.28 (Action composite).
L'action composite, pour les règles $\{R_1, \ldots R_m\}$ et une situation S, est applicable si :
– *C est cohérent, où $C = R_1 \overline{\boxtimes} \ldots \overline{\boxtimes} R_m$;*

[8] Le *discriminant maximal* est introduit, sous une forme légèrement différente dans (Sebag, 1996), pour concevoir un algorithme d'apprentissage inspiré de l'espace des versions (Mitchell, 1982).

Algorithme 4.27 : L'algorithme DAKAR 1 : *Discovery of Actionnable Knowledge And Recommendations*

Entrées : - un ensemble de règles $\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus}$
- une situation $\mathcal{S}$
- un paramètre M (le nombre d'actions souhaitées)

Sorties : - M actions valides, cohérentes et faisables

$\mathcal{A} = \bigcup_{R \in \mathcal{R}^{\oplus}} discr(\mathcal{S}, R)$ /*RQ : $\mathcal{A}$ contient des actions élémentaires valides*/
$faisceau1 = \{\}$
$faisceau2 = \{vrai\}$
tant que $faisceau1 \neq faisceau2$ **faire**
  $faisceau1 \leftarrow faisceau2$
  pour chaque $action1 \in faisceau1$ **faire**
    pour chaque $elem \in \mathcal{A}$ **faire**
      $action2 \leftarrow (action1 \wedge elem)$
      si $action2$ *cohérente et faisable* **alors**
        $faisceau2 \leftarrow faisceau2 \cup \{action2\}$
        si $card(faisceau2) > M$ **alors**
          $pire\_action \leftarrow Argmin_{A \in faisceau2} qualite(A)$
          $faisceau2 \leftarrow faisceau2 - \{pire\_action\}$

retourner faisceau1 (trié par rapport au critère de qualité)

– *et pour tout* i, $C[X_i] \not\subset S[X_i]$.

L'action composite est $composite(S, \{R_1, \ldots R_m\}) = (X_1 \in a_1) \wedge \ldots \wedge (X_n \in a_n)$ *où :*

$$a_i = \begin{cases} Dom_i & si\ S[X_i] \subseteq C[X_i] \\ d_i & sinon,\ avec\ C[X_i] \boxminus S[X_i] = \{d_i\} \end{cases}$$

Dans le deuxième cas, l'ensemble $C[X_i] \boxminus S[X_i]$ *est nécessairement de taille 1 (voir proposition 4.6).*

Une action composite est avant tout une action ; en particulier, la situation résultante d'une action composite appliquée à une situation est celle donnée dans la définition 4.16.

Exemple 4.29 (action composite).
Soient un attribut qualitatif X_l de domaine $\{a, b, c\}$, un attribut quantitatif X_t de domaine $[0, 10]$, une situation $S = (X_l \in \{a\}) \wedge (X_t \in [5, 5])$ et deux règles de classe $\oplus$: $R_1^{\oplus} = (X_l \in \{a, b\}) \wedge (X_t \in [5, 7]) \rightarrow \oplus$ et $R_2^{\oplus} = (X_t \in [6, 8]) \rightarrow \oplus$. On a alors :
$R_1^{\oplus} \overline{\boxtimes} R_2^{\oplus} = (X_l \in \{a, b\}) \wedge (X_t \in [6, 7])$, l'action composite est applicable et s'écrit :
$composite(S, \{R_1^{\oplus}, R_2^{\oplus}\}) = (X_l \in \{a, b, c\}) \wedge (X_t \in [6, 7])$
$res(S, composite(S, \{R_1^{\oplus}, R_2^{\oplus}\})) = (X_l \in \{a\}) \wedge (X_t \in [6, 7])$

Une action composite est une action de l'espace des actions tel que proposé dans la définition 4.24.

Proposition 4.30:
Si l'action $A = composite(S, \{R_1, \dots R_m\})$ est applicable alors elle est dans l'espace de recherche et $res(S, A)$ est couvert par les règles $\{R_1, \dots R_m\}$.

PREUVE. On montre que pour tout X_i, $A[X_i]$ ne contredit pas la définition de validité et $res(S, A)[X_i]$ est inclus dans $C[X_i] = R_1[X_i] \boxtimes \dots \boxtimes R_m[X_i]$.

- si $S[X_i] \subseteq C[X_i]$, $A[X_i] = Dom_i$ et ne peut pas contredire la définition de validité. On a aussi $res(S, A)[X_i] = S[X_i]$ et donc $res(S, A)[X_i] \subseteq C[X_i]$.
- sinon, comme également $C[X_i] \not\subset S[X_i]$, alors $A[X_i] = C[X_i] \boxminus S[X_i] = \{d\}$ (proposition 4.6). Par construction avec l'opérateur $\boxminus$ (voir définition 4.4), on a $d \boxtimes S[X_i] = \emptyset$ donc $A[X_i]$ ne contredit pas la validité. On a aussi $d \subseteq C[X_i]$ et $res(S, A)[X_i] = d$ donc $res(S, A)[X_i] \subseteq C[X_i]$.

L'algorithme présenté dans cette sous-section s'appuie sur l'élaboration d'actions composites. Afin de prouver l'optimalité de la recherche lorsque certaines conditions sont remplies, nous utilisons les propriétés suivantes.

Proposition 4.31:
Si l'action $A = composite(S, \{R_1, \dots R_m\})$ n'est pas applicable, alors il n'existe pas d'action A' de l'espace de recherche telle que $res(S, A')$ soit couvert par toutes les règles $\{R_1, \dots R_m\}$.

PREUVE. Supposons que A' existe, comme A n'est pas applicable, on est dans un des deux cas suivants :

- $C = R_1 \overline{\boxtimes} \dots \overline{\boxtimes} R_m$ n'est pas cohérent. Alors il existe un attribut X_i tel que $C[X_i] = \emptyset$. Si on suppose que $res(S, A')$ est couvert par les règles $\{R_1, \dots R_m\}$, alors $res(S, A')[X_i] \subseteq C[X_i]$ et donc est inclus dans $\emptyset$. L'action A' ne peut pas être cohérente.
- il existe X_i tel que $C[X_i] \subset S[X_i]$. Si A' existe alors $res(S, A')[X_i] \subseteq C[X_i]$ et $res(S, A')[X_i] \subset S[X_i]$. Donc A' est une précision de S sur l'attribut X_i. L'action A' n'est pas valide.

Proposition 4.32:
Si la faisabilité est monotone et l'action $A = composite(S, \{R_1, \dots R_m\})$ est applicable mais non faisable alors il n'existe pas d'action A' faisable telle que $res(S, A')$ soit couvert par les règles $\{R_1, \dots R_m\}$.

PREUVE. Supposons que A' existe. Puisque la faisabilité est monotone, si on montre que pour tout X_i, $A'[X_i] \subseteq A[X_i]$ alors comme A n'est pas faisable, A' ne doit pas être faisable puisque A couvre A'. Pour tout i, on se trouve dans un des cas suivants (où $C = R_1 \overline{\boxtimes} \dots \overline{\boxtimes} R_m$) :

- $S[X_i] \subseteq C[X_i]$, on a $A[X_i] = Dom_i$ et nécessairement $A'[X_i] \subseteq A[X_i]$.
- sinon, on a avec la propriété 4.6 que $C[X_i] \boxminus S[X_i]$ est de taille 1. On note $A[X_i] = d$. On ne peut pas avoir $A'[X_i] = Dom_i$. En effet, sinon on aurait $res(S, A')[X_i] = S[X_i]$, d'où $res(S, A')[X_i] \not\subseteq C[X_i]$ et $res(S, A')$ n'est pas couvert

par C. Pour que A' soit valide, il faut alors que $A'[X_i] \boxtimes S[X_i] = \emptyset$, et par construction $res(S, A')[X_i] = A'[X_i]$. On doit donc avoir à la fois $A'[X_i] \boxtimes C[X_i] = A'[X_i]$ (pour que C couvre $res(S, A')$) et $A'[X_i] \boxtimes S[X_i] = \emptyset$; d'après la définition de $\boxminus$, il existe $e \in C[X_i] \boxminus S[X_i]$ tel que $A'[X_i] \subseteq e$, $e = d$ et $A'[X_i] \subseteq A[X_i]$.

Les ensembles de règles que nous considérons lors de l'élaboration d'une action composite sont en fait les cliques d'un graphe de règles.

Définition 4.33 (Graphe et clique de règles).
Soit $R = \{R_1, \ldots R_m\}$ un ensemble de règles, le graphe (non dirigé) de R, noté graphe(R) est un graphe où chaque nœud est une règle de R et où il existe une arête entre R_i et R_j si R_i et R_j sont connectées. Une clique de graphe(R) est un sous-ensemble de R où les règles sont deux à deux connectées.

Soit un graphe de règles G, $cliques\_max(G)$ est l'ensemble des cliques maximales. Pour une clique c, $cliques\_inf(c)$ est l'ensemble des cliques contenues dans c et de taille $card(c) - 1$. On note enfin $som\_conf(c)$ la somme des confiances des règles de c.

Exemple 4.34 (cliques de règles et actions composites).
Soient un attribut qualitatif X_l de domaine $\{a, b, c\}$, un attribut quantitatif X_t de domaine $[0, 10]$, une situation $S = (X_l \in \{b\}) \wedge (X_t \in [3, 3])$ et quatre règles dont le graphe est illustré dans la figure 4.35. Ce graphe G contient deux cliques maximales :
$cliques\_max(G) = \{\{R_1^{\oplus}, R_2^{\oplus}, R_3^{\oplus}\}, \{R_2^{\oplus}, R_4^{\oplus}\}\}$
Si on considère la première clique, on a :
$R_1^{\oplus} \overline{\boxtimes} R_2^{\oplus} \overline{\boxtimes} R_3^{\oplus} = (X_l \in \{a\}) \wedge (X_t \in [4, 5])$
$composite(S, \{R_1^{\oplus}, R_2^{\oplus}, R_3^{\oplus}\}) = (X_l \in \{a\}) \wedge (X_t \in [4, 5])$
$res(S, composite(S, \{R_1^{\oplus}, R_2^{\oplus}, R_3^{\oplus}\})) = (X_l \in \{a\}) \wedge (X_t \in [4, 5])$
Si on considère la deuxième clique, on a :
$R_2^{\oplus} \overline{\boxtimes} R_4^{\oplus} = (X_l \in \{a, b\}) \wedge (X_t \in [7, 8])$
$composite(S, \{R_2^{\oplus}, R_4^{\oplus}\}) = (X_l \in \{a, b, c\}) \wedge (X_t \in [7, 8])$
$res(S, composite(S, \{R_2^{\oplus}, R_4^{\oplus}\})) = (X_l \in \{b\}) \wedge (X_t \in [7, 8])$
D'autres actions composites existent, par exemple à partir des cliques de :
$cliques\_inf(\{R_1^{\oplus}, R_2^{\oplus}, R_3^{\oplus}\}) = \{\{R_1^{\oplus}, R_2^{\oplus}\}, \{R_1^{\oplus}, R_3^{\oplus}\}, \{R_2^{\oplus}, R_3^{\oplus}\}\}$

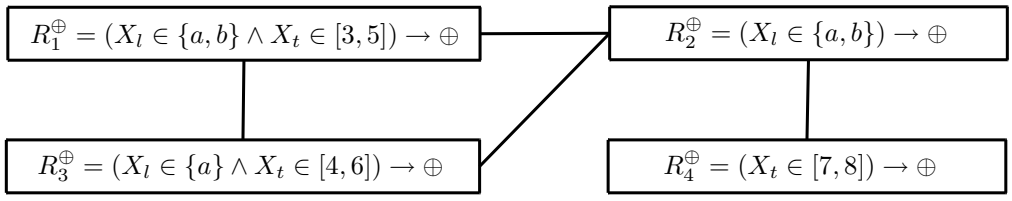


FIG. 4.35 – Illustration du graphe des règles $R_1^{\oplus}$, $R_2^{\oplus}$, $R_3^{\oplus}$ et $R_4^{\oplus}$. Il contient 2 cliques maximales : $\{R_1^{\oplus}, R_2^{\oplus}, R_3^{\oplus}\}$ et $\{R_2^{\oplus}, R_4^{\oplus}\}$.

Dans certaines conditions, on peut être sûr qu'une action composite maximise le critère de qualité dans l'espace des actions.

Proposition 4.36:
Soit S une situation, $\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus}$ un ensemble de règles et c une clique de $graphe(\mathcal{R}^{\oplus})$ telle que l'action $A = composite(S, c)$ est applicable et faisable, alors A est optimale dans l'espace de recherche (c'est-à-dire une action qui maximise le critère de qualité) si les conditions suivantes sont remplies :

1. $res(S, A)$ n'est couvert par aucune règle de $\mathcal{R}^{\ominus}$;
2. pour toute clique c' de $graphe(\mathcal{R}^{\oplus})$ telle que $som\_conf(c') > qualite(S, A)$, l'action $composite(S, c')$ est soit non applicable, soit non faisable ;
3. la faisabilité est monotone.

PREUVE. Comme $res(S, A)$ n'est couvert par aucune règle de $\mathcal{R}^{\ominus}$, la propriété 4.30 permet de déduire que $qualite(S, A) \geq som\_conf(c)$. Supposons qu'il y ait une action A' faisable dans l'espace de recherche telle que $qualite(S, A') > qualite(S, A)$. Soit c' la clique des règles couvrant $res(S, A')$, on a $som\_conf(c') \geq qualite(S, A')$ par la définition de qualité et $som\_conf(c') \geq qualite(S, A') > qualite(S, A) \geq som\_conf(c)$. D'après la condition 2, on est dans un des deux cas suivants :

- soit l'action $composite(S, c')$ n'est pas applicable : dans ce cas il n'existe pas d'actions B telle que $res(S, B)$ soit couvert par toutes les règles c' (propriété 4.31). Donc A' n'existe pas ;
- soit l'action $composite(S, c')$ n'est pas faisable : étant donné que la faisabilité est monotone, toute action B telle que $res(S, B)$ soit couvert par les règles de c', est infaisable (propriété 4.32). Donc A' est infaisable.

L'algorithme 4.37 s'appuie sur la propriété 4.36 pour trouver l'optimum dans le cas où les conditions sont remplies. Il maintient l'ensemble des cliques $\mathcal{C}$ du graphe de $\mathcal{R}^{\oplus}$ pour lesquelles l'action composite n'a pas encore été construite. Au début, il considère les cliques maximales Cls qui maximisent la valeur som_conf. Si pour une de ces cliques l'action composite est applicable, faisable et définie de sorte que la situation résultante ne soit couverte par aucune règle de $\mathcal{R}^{\ominus}$, alors l'action est un optimum global. Si au contraire, il existe une action composite applicable et faisable mais telle que la situation résultante soit couverte par une règle de $\mathcal{R}^{\ominus}$, alors on ne peut pas conclure que les actions proposées contiennent un optimum global. Enfin, si aucune action composite applicable n'est faisable, alors on peut s'intéresser aux cliques suivantes : celles qui maximisent som_conf dans l'ensemble des cliques du graphe en ayant préalablement retiré les cliques Cls de $\mathcal{C}$. Une des conditions requises pour que l'algorithme construise un optimum global est que la faisabilité soit monotone. Si la faisabilité n'est pas monotone, on ne peut pas savoir si une action construite est un optimum.

4.2.6 Expérimentations

Dans ces expérimentations, nous souhaitons comparer les deux algorithmes proposés dans les sous-sections 4.2.4 et 4.2.5 que nous identifierons respectivement par $dak1$ et

Algorithme 4.37 : L'algorithme DAKAR 2

Entrées : - un ensemble de règles $\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus}$
- une situation $\mathcal{S}$
- un paramètre M (le nombre d'actions souhaitées)

Sorties : - un ensemble d'actions cohérentes, valides et faisables
- un booléen *optimum_trouve*

$\mathcal{C} \leftarrow cliques\_max(graphe(\mathcal{R}^{\oplus}))$
$action = \{vrai\}$
tant que $(card(actions) \leq M)$ *et* $(card(\mathcal{C}) > 0)$ **faire**
  $Cls \leftarrow \{cl \in \mathcal{C} | som\_conf(cl) = max_{c \in \mathcal{C}}(som\_conf(c))\}$
  $actions\_locales = \{\}$
  $optimum\_local\_trouve = faux$
  pour chaque $clique \in Cls$ **faire**
    $ajouter\_sous\_cliques = vrai$
    si $composite(S, clique)$ *est applicable et faisable* **alors**
      $actions\_locales \leftarrow actions\_locales \cup \{composite(S, clique)\}$
      si $\forall r \in \mathcal{R}^{\ominus}$, *r ne couvre pas* $res(composite(S, clique), S)$ **alors**
        $ajouter\_sous\_cliques = faux$
        $optimum\_local\_trouve = vrai$
    si $ajouter\_sous\_cliques$ **alors**
      $\mathcal{C} = \mathcal{C} \cup cliques\_inf(c)$
  si $(actions = \{vrai\})$ *et* $(card(actions\_locales) > 0)$ **alors**
    $optimum\_trouve = optimum\_local\_trouve$
  $actions \leftarrow actions \cup actions\_locales$
  $\mathcal{C} \leftarrow \mathcal{C} - Cls$
si *la faisabilité n'est pas monotone* **alors**
  $optimum\_trouve = false$
retourner *actions* et *optimum_trouve*

*dak*2. Pour cela nous utilisons la théorie induite avec le système CN2 pour la caractérisation du transfert d'herbicides pour les arbres d'exutoires (voir section 3.5). La théorie est composée de 17 règles de classe $\oplus$ ($transfert\_faible$) et de 16 règles de classe $\ominus$ ($transfert\_important$). Le graphe des règles de classe $\oplus$ est composé de 28 cliques maximales, et d'un total de 393 cliques.

L'application comporte 11 attributs dont 9 sont quantitatifs (voir tableau 3.23). Le tableau 4.38 donne pour chaque attribut son poids de stabilité (définition 4.19) et indique s'il est flexible (définition 4.18). De manière générale, les attributs relatifs aux pratiques agricoles sont plus flexibles que ceux relatifs à l'occupation du sol ou à la topologie.

Nom de l'attribut	**Poids de stabilité**	**Flexibilité**
agr_rapport_risque_faible	0.03	oui
agr_rapport_risque_fort	0.03	oui
agr_pression	0.03	oui
agr_rapport_prelevee	0.03	oui
agr_rapport_disp_tampon	0.03	oui
agr_surf_max_mais	0.07	**?**
agr_rapport_surf_mais	0.08	non
agr_topo_forme_largeur	0.10	non
agr_surface	0.10	non
agr_max_profondeur	0.25	non
agr_topo_forme_denivele	0.25	non

TAB. 4.38 – Flexibilité des attributs pour la faisabilité. L'attribut $agr\_surf\_max\_mais$ est jugé flexible dans l'expérience $exp1.a$ et non flexible dans l'expérience $exp1.b$.

La recommandation d'actions a été réalisée pour les 587 exemples tests de classe $transfert\_important$ de la base d'apprentissage exposée en section 3.3. Ces exemples sont considérées comme des situations insatisfaisantes.

Nous avons défini deux plans d'expérience :

- **exp1.a (exp1.b)** : nous nous intéressons ici au paramètre M (nombre d'actions souhaitées), il varie de 1 à 5. La définition de faisabilité utilisée est la faisabilité par restriction (définition 4.18). Dans l'expérience **exp1.a** (respectivement **exp1.b**), l'attribut $agr\_surf\_max\_mais$ (surface de l'exutoire cultivé en maïs le plus grand) est flexible (respectivement non flexible). Pour chacune des valeurs du paramètre M et chacune des 587 situations insatisfaisantes, une recommandation d'actions est effectuée à l'aide des algorithmes *dak*1 et *dak*2.
- **exp2** : nous nous intéressons ici au paramètre δ lorsque la définition de faisabilité utilisée est la δ-faisabilité (définition 4.19). Nous faisons varier δ dans l'ensemble $\{0.01, 0.02, 0.04, 0.08, 0.15, 0.3\}$, le nombre d'actions souhaitées est fixé à 3. Pour chacune des valeurs de δ et chacune des 587 situations insatisfaisantes, la recom-

mandation de 3 actions s'effectue à l'aide des algorithmes $dak1$ et $dak2$.

Avant de comparer les résultats pour les différents plans d'expérience, nous donnons deux exemples de résultats. Dans l'exemple 4.39, la faisabilité utilisée est la faisabilité par restriction, un optimum global est construit avec l'algorithme $dak2$. Dans l'exemple 4.40, on ne peut pas savoir si une action proposée est un optimum global car la faisabilité utilisée est la δ-faisabilité qui est non monotone.

Exemple 4.39 (actions recommandées pour le plan d'expérience exp1).
Nous considérons la situation S décrite par les attributs agrégats :
`agr_pression=1446` g
`agr_rapport_risque_faible=100 %`
`agr_rapport_risque_fort=0 %`
`agr_rapport_prelevee=0 %`
`agr_surface=32800` m^2
`agr_rapport_surf_mais=98 %`
`agr_surf_max_mais=32000` m^2
`agr_rapport_disp_tampon=0 %`
`agr_max_profondeur=2 exutoires`
`agr_topo_forme_denivele=plate`
`agr_topo_forme_largeur=i`
Les actions recommandées par l'algorithme $dak1$ dans le cadre de l'expérience $exp1.a$ sont les suivantes :
`agr_pression` $\in$ `[6,181.5[ (qualité=0)`
`agr_pression` $\in$ `[6,83[ (qualité=0)`
Ces deux actions sont assez proches finalement, il s'agit d'une proposition de diminuer, de beaucoup, la quantité de pesticides appliquée. Les actions recommandées par l'algorithme $dak2$ dans le cadre de l'expérience $exp1.a$ sont les suivantes (contenant un optimum) :
`vrai (qualité=-2)`
`agr_rapport_disp_tampon` $\in$ `]5.5,67]` $\wedge$ `agr_surf_max_mais` $\in$ `[400,4200[ (qualité=1)`
L'action $vrai$ signifie aucune modification des valeurs des attributs. L'action optimale construite suggère d'ajouter un dispositif tampon et de diminuer de manière importante la surface de l'exutoire cultivé en maïs le plus important. Les actions proposées consistent en des modifications importantes de la valeur des attributs. L'utilisation de δ-faisabilité, au lieu de la faisabilité par restriction, permet justement d'éviter cela (voir exemple 4.40). Dans le cadre du plan $exp1.b$, où l'attribut $agr\_surf\_max\_mais$ n'est pas flexible, $dak1$ propose les mêmes actions et $dak2$ ne propose finalement que l'action $vrai$ pour la situation S.

Exemple 4.40 (actions recommandées pour le plan d'expérience exp2).
Nous considérons la situation S décrite par les attributs agrégats :
`agr_pression=256` g
`agr_rapport_risque_faible=61 %`
`agr_rapport_risque_fort=39 %`
`agr_rapport_prelevee=39 %`

`agr_surface=56000` m^2
`agr_rapport_surf_mais=9 %`
`agr_surf_max_mais=2000` m^2
`agr_rapport_disp_tampon=0 %`
`agr_max_profondeur=5 exutoires`
`agr_topo_forme_denivele=concave`
`agr_topo_forme_largeur=v`

Dans le cadre du plan $exp2$ avec le paramètre $\delta = 0.01$, l'action de qualité la plus importante proposée par $dak1$ est la suivante :

`agr_rapport_risque_fort` $\in$ `[0,39[` $\wedge$ `agr_surface` $\in$ `]42200,53800[ (qualité=1)`

L'action de qualité la plus importante proposée par $dak2$ pour le plan $exp2$ est la suivante :

`agr_rapport_surf_mais` $\in$ `[0,6.5[` $\wedge$ `agr_surf_max_mais` $\in$ `[400,1000[` $\wedge$
`agr_surface` $\in$ `]31200,53800[ (qualité=1.99)`

Les actions proposent de modifier l'attribut $agr\_surface$ qui est un attribut plutôt stable (voir tableau 4.38). Mais elles consistent en de faibles modifications de la valeur de cet attribut, et ceci grâce à l'utilisation de la δ-faisabilité. De manière générale, ces deux actions suggèrent une combinaison de petites modifications (liées aux seuils dans les valeurs des attributs quantitatifs). Celles proposées par la première action concernent le risque du pesticide utilisé et la surface totale ; celles proposées par la seconde concernent l'utilisation de la surface pour la culture du maïs, la surface maximale cultivée en maïs et la surface totale.

Les critères de comparaison des algorithmes $dak1$ et $dak2$ sont le temps moyen de la recommandation d'actions, le nombre moyen d'actions construites (*Constructions*) et la *Performance*. La performance consiste en fait à comparer les algorithmes sur le nombre de fois où l'un des deux propose une action dont le critère de qualité est plus important que pour le deuxième.

À l'aide de $dak2$, un optimum global est construit dans 419 cas sur 587 pour le plan **exp1.a** et dans 123 cas pour le plan **exp1.b**, quel que soit M. Par construction, quel que soit M, l'action proposée par $dak2$ au critère de qualité le plus important est la même.

Param.	**Temps moyen**[6]		**Constructions**		**Performance**[7]	
M	$dak1$	$dak2$	$dak1$	$dak2$	$dak1 > dak2$	$dak1 < dak2$
1	291	77	235.96	354.08	6.3%	40.03%
2	495	80	408.09	383.97	6.3%	36.12%
3	705	81	581.47	388.06	6.3%	34.24%
4	982	82	789.86	390.52	6.47%	32.54%
5	1211	83	996.93	392.14	6.98%	31.18%

TAB. 4.41 – Résultats du plan d'expérience **exp1.a**

Les résultats des recommandations d'actions issues du plan d'expériences **exp1.a** sont résumées dans le tableau 4.41. Les temps moyens et le nombre d'actions construites augmentent rapidement avec M pour l'algorithme $dak1$. Pourtant la qualité de la meilleure action proposée par cet algorithme n'augmente pas avec M relativement à la qualité constante de la meilleure action proposée par $dak2$: la performance de $dak1$ est améliorée de seulement 0.68% lorsque que le paramètre M passe de 1 à 5.

Param.	**Temps moyen**		**Constructions**		**Performance**	
M	$dak1$	$dak2$	$dak1$	$dak2$	$dak1 > dak2$	$dak2 > dak1$
1	204	78	168.55	389.2	48.38%	10.39%
2	345	78	293.02	392.67	48.38%	8.18%
3	498	78	421.21	392.93	48.38%	8.18%
4	681	78	581.51	393	48.55%	8.01%
5	891	79	751.74	393	48.55%	8.01%

TAB. 4.42 – Résultats du plan d'expérience **exp1.b**.

Les résultats des recommandations d'actions issues du plan d'expériences **exp1.b** sont résumées dans le tableau 4.42. La différence réside dans la faisabilité : ici, une action proposant une modification de la valeur de l'attribut $agr\_surf\_max\_mais$ n'est pas faisable, la faisabilité est donc plus restrictive. Alors que les temps moyens et les nombres d'actions élaborées pour l'algorithme $dak1$ évoluent de la même façon avec M que précédemment, la performance de cet algorithme est cette fois-ci plus importante que celle de $dak2$. Cela peut s'expliquer par les stratégies des algorithmes. La stratégie de $dak1$ est l'ajout successif de sélecteurs à des actions préalablement construites et dont l'évaluation est suffisamment bonne pour que l'action soit conservée. Ceci peut permettre, sinon de garantir que la situation résultante soit couverte par un nombre important de règles de classe $\oplus$, au moins que la situation résultante ne soit pas couverte par trop de règles de classe $\ominus$.

En définitif, $dak1$ est plus adapté à un cas où la faisabilité est sévère, c'est à dire quand peu d'actions sont faisables. Dans le cas d'une faisabilité souple (où l'on s'accorde à effectuer des modifications importantes), l'algorithme $dak2$, pour la recherche d'un optimum, est plus efficace. Dans notre application, pour une faisabilité par restriction, l'attribut $agr\_surf\_max\_mais$ constitue un attribut clé pour la recommandation d'actions dans le sens où, si l'on accepte des actions suggérant de modifier la valeur de cet attribut alors $dak2$ est mieux adapté que $dak1$, sinon c'est le contraire.

Le plan d'expérience **exp2** montre cela lorsque l'on utilise la δ-faisabilité en faisant croître le paramètre δ. Plus δ est petit, plus la faisabilité est sévère. On voit que la performance de $dak2$ augmente avec δ (tableau 4.43).

[6] les expériences ont été effectuées sur un Intel Xeon 3.4GHz et les temps sont millisecondes.

[7] Le reste du temps les meilleures actions proposées par $dak1$ et $dak2$ sont de qualité égale.

Param.	Temps moyen		Constructions		Performance	
δ	$dak1$	$dak2$	$dak1$	$dak2$	$dak1 > dak2$	$dak2 > dak1$
0.01	652	120	494.53	390.3	46%	14.14%
0.02	798	116	612.98	368.45	28.79%	38.67%
0.04	1131	94	845.84	252.84	6.47%	61.33%
0.08	1592	45	1154.5	49.41	0.68%	78.36%
0.15	1665	31	1197.61	4.62	0.34%	97.1%
0.3	1683	30	1204.54	3	0%	100%

TAB. 4.43 – Résultats du plan d'expérience **exp2**

4.3 Visualisation des règles

L'outil présenté dans cette section est une interface de visualisation en deux parties, celle où apparaissent les exemples et celle où apparaissent les règles. Le but est de pouvoir faire la liaison entre certains exemples et certaines règles, en s'appuyant sur la relation de couverture.

Nous considérons deux ensembles d'objets $AllE$ et $AllR$ qui sont respectivement l'ensemble des exemples et des règles utilisés pour la visualisation, chaque objet possédant un identifiant unique. L'objectif est de pouvoir se focaliser sur certains objets d'intérêt pour l'utilisateur ; ceux-ci sont stockés dans les ensembles $SelE$ (pour les exemples) et $SelR$, qui, au début du processus de visualisation, sont vides. Ainsi, l'utilisateur peut être intéressé par les exemples qui sont couverts par une règle particulière. En pratique, la visualisation doit alors faire ressortir ces exemples.

Le processus de visualisation consiste en une suite de requêtes de l'utilisateur, chaque requête permettant de déterminer soit l'ensemble d'exemples d'intérêt $SelE$, soit l'ensemble des règles d'intérêt $SelR$. Ces requêtes doivent être formulées dans un langage dont voici la grammaire :

```
1  ExpR -> union_r(ExpR,ExpR)
2        | inter_r(ExpR,ExpR)
3        | diff_r(ExpR,ExpR)
4        | covering_all(ExpE)
5        | covering_some(ExpE)
6        | pattern_r(ExprR,Pattern)
7        | all_r
8        | selected_r
9        | Pattern
10 ExpE -> union_e(ExpE,ExpE)
11       | inter_e(ExpE,ExpE)
12       | diff_e(ExpE,ExpE)
13       | covered_by_all(ExpR)
14       | covered_by_some(ExpR)
15       | pattern_e(ExprE,Pattern)
```

```
16       | all_e
17       | selected_e
18       | Pattern
19 Pattern -> <String>
```

Une requête valide pour la sélection des exemples $SelE$ (respectivement des règles $SelR$) est une phrase respectant la grammaire et dont la première règle de dérivation de la grammaire appliquée est une règle $ExpE$ (respectivement $ExpR$), c'est-à-dire une règle numérotée entre 1 et 9 (respectivement entre 10 et 18).

Exemple 4.44 (requête valide).
La requête `covering_some(union_e("ex1","ex2"))` est valide pour la sélection d'un ensemble de règles, la figure illustre les dérivations à effectuer pour la construction de cette requête.

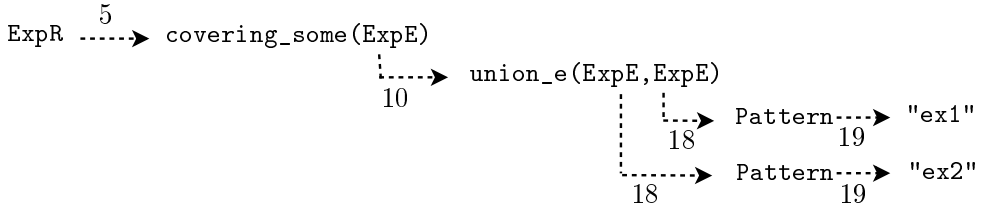


FIG. 4.45 – Règles de dérivation (illustrées par les flèches) utilisées afin de construire la requête `covering_some(union_e("ex1","ex2"))` qui est valide pour la sélection de règles de classification.

Le calcul de l'ensemble $SelE$ ou $SelR$ se fait alors, à partir d'une requête valide, à l'aide de la fonction $calcE$ ou $calcR$. Soit r, $r1$ et $r2$ trois requêtes valides pour la sélection des règles ; soient e, $e1$ et $e2$ trois requêtes valides pour la sélection des exemples et soit p un motif pour la sélection ; les fonctions $calcR$ et $calcE$ sont définies[9] par une double récursion :

$$calcR(r) = \begin{cases} calcR(r1) \cup calcR(r2) & \text{si } r = union\_r(r1, r2) \\ calcR(r1) \cap calcR(r2) & \text{si } r = inter\_r(r1, r2) \\ calcR(r1) - calcR(r2) & \text{si } r = diff\_r(r1, r2) \\ \{ru \in AllR | calcE(e) \subseteq covers(ru)\} & \text{si } r = covering\_all(e) \\ \{ru \in AllR | calcE(e) \cap covers(ru) \neq \emptyset\} & \text{si } r = covering\_some(e) \\ \{ru \in calcR(r1) | match(ru, p)\} & \text{si } r = pattern(r1, p) \\ AllR & \text{si } r = all\_r \\ SelR & \text{si } r = selected\_r \\ calcR(pattern\_r(all\_r, p)) & \text{si } r = p \end{cases}$$

[9]Les symboles $\cap$, $\cup$ et $-$ représentent l'intersection, l'union et la différence ensembliste. La fonction $match(o, p)$, où o est un identifiant d'objet (règle ou exemple) et p un motif, est à vrai si o respecte l'expression régulière p. Les symboles ex et ru désignent un identifiant d'exemple et de règle quelconques.

$$calcE(e) = \begin{cases} calcE(e1) \cup calcE(e2) & \text{si } e = union\_e(e1, e2) \\ calcE(e1) \cap calcE(e2) & \text{si } e = inter\_e(e1, e2) \\ calcE(e1) - calcR(e2) & \text{si } e = diff\_e(e1, e2) \\ \{ex \in AllE | \forall ru \in calcR(r), ex \in covers(ru)\} & \text{si } e = covered\_by\_all(r) \\ \{ex \in AllE | \exists ru \in calcR(r), ex \in covers(ru)\} & \text{si } e = covered\_by\_some(r) \\ \{ex \in calcE(e1) | match(ex, p)\} & \text{si } e = pattern(e1, p) \\ AllE & \text{si } e = all\_e \\ SelE & \text{si } e = selected\_e \\ calcE(pattern\_e(all\_e, p)) & \text{si } e = p \end{cases}$$

Lorsque qu'une requête est définie par l'utilisateur, les ensembles d'objets sélectionnés sont mis à jour et la visualisation est modifiée pour se focaliser sur la sélection. Cette mise à jour, à chaque requête, de la sélection permet un calcul incrémental des objets sur lesquels focaliser la visualisation. Ceci afin d'éviter la saisie de requêtes trop complexes. L'exemple 4.46 présente quelques requêtes types.

Exemple 4.46 (requêtes valides de sélection d'exemples ou de règles).
Voici quelques requêtes valides pour la sélection d'exemples et l'ensemble des exemples sélectionnés $SelE$ correspondant :

- `pattern_e(all_e,"class1"))` : l'ensemble des exemples dont l'identifiant respecte l'expression régulière "class1". Dans notre application, on peut, à l'aide de cette requête, faire ressortir les arbres d'exutoires de classe $transfert\_important$;
- `diff_e(covered_by_all("ru1"),covered_by_all("ru2"))` : l'ensemble des exemples couverts par "ru1" mais non couverts par "ru2" ;

Voici quelques exemples de requêtes valides pour la sélection de règles et l'ensemble de règles sélectionnées $SelR$ correspondant :

- `covering_some(selected_e)` : l'ensemble des règles couvrant au moins un exemple sélectionné. Ainsi, si on sélectionne à la main un arbre d'exutoire de classe $transfert\_important$ sur lequel il existe une zone de tampon, on peut voir les règles expliquant la classe de cet exemple, qui est inattendue ;
- `diff_r(covering_all("class1"),covering_some("class2"))` : l'ensemble des règles couvrant tous les exemples "class1" et aucun exemple "class2".

Dans notre application, les exemples sont intégrés dans une visualisation basée sur une carte du bassin versant comprenant le réseau hydrographique (à gauche dans la figure 4.47). Les exemples sélectionnés se démarquent des autres par leur couleur. La couleur est rouge (respectivement verte) si l'exemple est de classe $transfert\_important$ (respectivement $transfert\_faible$). Les règles et motifs sélectionnés apparaissent quant à eux sur la partie droite de l'interface.

La figure 4.47 illustre un exemple de processus de visualisation en deux étapes. L'utilisateur sélectionne deux règles puis il soumet la requête `covered_by_some(selected_r)` pour la sélection des exemples. Les arbres d'exutoires sélectionnés ressortent sur une carte du bassin versant grâce à leur couleur qui dépend de leur classe.

L'interface peut également être utilisée pour la recommandation d'actions. Ainsi, l'utilisateur sélectionne un arbre d'exutoires, à la main où à l'aide d'une requête, puis

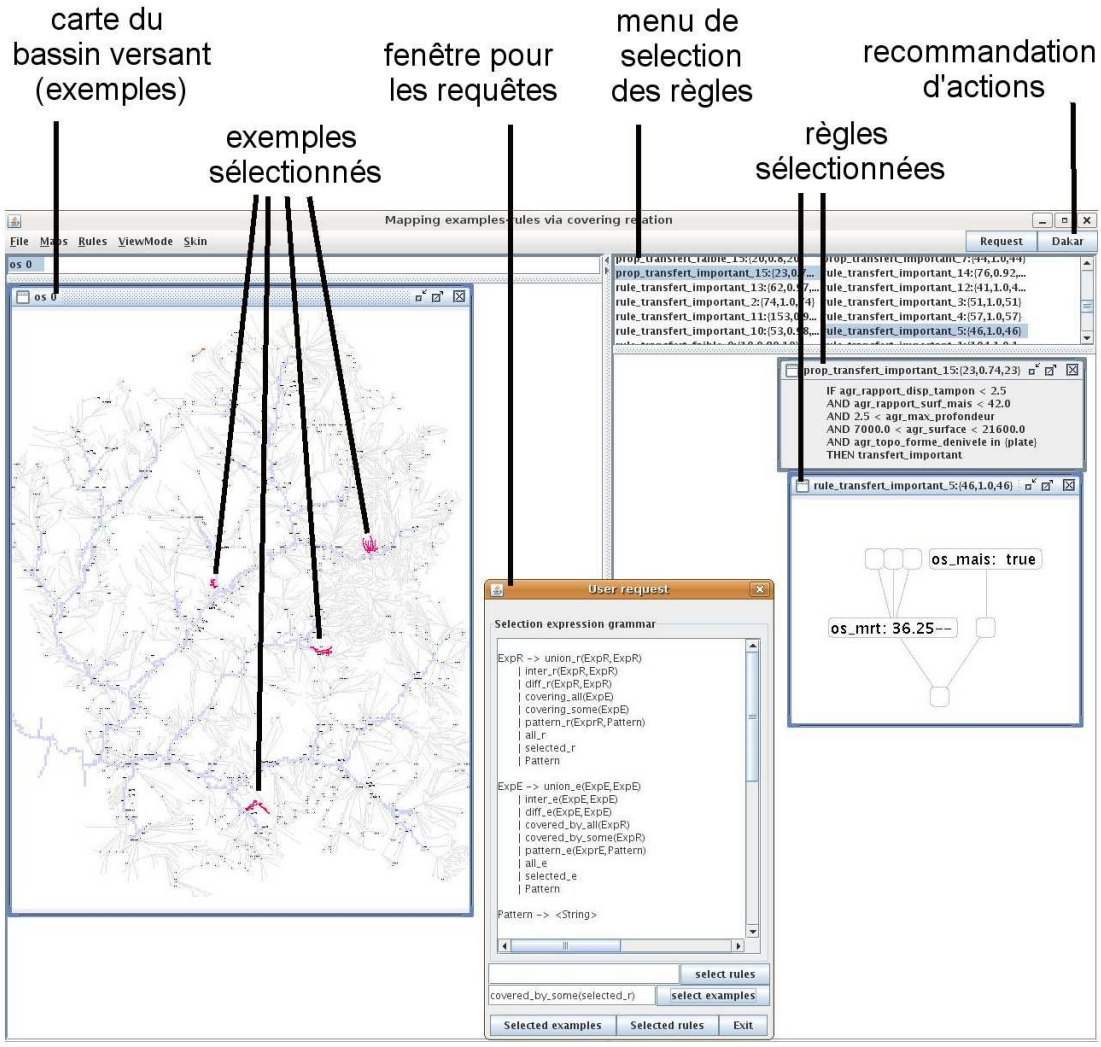


FIG. 4.47 – Exemple de processus de visualisation en deux étapes. Deux règles (à droite) sont tout d'abord sélectionnées par l'utilisateur : une règle en logique attribut-valeur (section 3.5) et un motif d'arbre d'exutoires (section 3.4). La deuxième étape consiste en la sélection des exemples couverts par au moins une de ces règles. La requête tapée est `covered_by_some(selected_r)` (fenêtre du milieu) et les arbres d'exutoires couverts par une des deux règles ressortent alors sur une carte du bassin versant (en rouge car ils sont de classe $transfert\_important$).

recquiert des actions qui sont suggérées par les algorithmes DAKAR. Les règles sur lesquelles s'appuient la recommandation d'action sont les règles, en logique attribut-valeur, sélectionnées.

4.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'analyse, orientée aide à la décision, d'un ensemble de règles.

Nous avons tout d'abord présenté une tâche originale de recommandation d'actions afin d'améliorer une situation défavorable à l'utilisateur, ceci en s'appuyant sur la théorie. Un espace de recherche des actions a été défini. Deux algorithmes sont proposés pour réaliser cette tâche. Le premier repose sur une recherche par faisceau des actions au cours de laquelle les actions sont spécialisées (Trépos *et al.*, 2005, 2006). Il spécialise les actions par ajouts successifs de sélecteurs, l'espace de recherche est donc parcouru à l'aide de "petites" spécialisations. Le second algorithme construit dans certaines conditions un optimum de l'espace des actions en combinant plusieurs règles. Il construit directement des actions complexes.

La comparaison des algorithmes montre que selon la définition de faisabilité d'une action que l'on utilise, l'un peut produire des actions de plus grande qualité que l'autre, ou inversement. L'étude comparative s'est appuyée sur une théorie en logique attribut-valeur induite dans le cadre du projet SACADEAU (chapitre 3). Il serait intéressant de combiner les deux algorithmes. Ainsi, on peut construire des actions complexes puis leur ajouter des sélecteurs afin de rechercher des actions proches de l'action complexe. En les combinant, on pourrait espérer de meilleurs résultats.

Un outil de visualisation des règles a également été proposé dans ce chapitre. Nous nous sommes particulièrement intéressés à la relation de couverture entre les règles et les exemples. Pour cela, nous avons utilisé un langage de requête. L'exécution d'une requête du langage implique la sélection d'exemples ou de règles qui ressortent visuellement sur l'interface. Cet outil intègre également les méthodes de recommandation d'actions développées. Dans le cadre de notre application, cette interface nous permet ainsi de localiser les parties de bassin versant, en fait les arbres d'exutoires, qui sont concernées par des règles. On peut également connaître les explications, exprimées sous forme de règles, de l'importance du transfert d'un arbre d'exutoires. Enfin, elle permet de suggérer des actions pour une diminution du transfert.

Conclusion

Dans cette thèse, nous nous sommes intéressés à l'analyse des données de simulation issues d'un modèle représentant le fonctionnement d'un système environnemental. Ces modèles sont en général complexes car ils couplent différents modèles. Leurs entrées et sorties peuvent être nombreuses et exprimées sous plusieurs formes. Il s'agit par exemple de variables qualitatives ou quantitatives, de relations spatiales et temporelles. L'approche proposée dans ce travail repose sur des techniques d'apprentissage symbolique afin d'analyser les données de simulation. En particulier, nous nous sommes focalisés sur l'apprentissage de règles reliant les entrées et les sorties du modèle de notre application. L'application concerne le transfert de pesticides au sein d'un bassin versant. Les objectifs étaient, en s'appuyant sur l'analyse des règles induites, d'une part de dégager les variables explicatives des sorties de simulation, d'autre part de suggérer des actions permettant d'améliorer une situation, par exemple dans un cas de contamination par les pesticides.

Au sein du projet SACADEAU, un modèle de simulation du transfert de pesticides, à l'échelle d'un bassin versant et d'une saison culturale, a été développé. Il repose sur le couplage d'un modèle décisionnel et d'un modèle biophysique. Le premier simule les pratiques agricoles ; il permet en particulier de déterminer, lors de la simulation, quand et où sont appliqués les pesticides. Le second permet la simulation du transfert de pesticides, depuis son lieu d'application jusqu'au réseau hydrographique. Le modèle, développé en amont de cette thèse, possède une représentation des transferts de ruissellement reposant sur une structure d'arbre. Cette représentation constitue une modélisation de la connexion des flux entre parcelles agricoles.

Contributions

Dans le cadre de l'apprentissage automatique de règles pour notre application, nous avons constitué une base d'exemples étiquetés ; chacun représenté par un arbre où les nœuds sont décrits par des attributs. Deux approches ont été proposées pour l'induction des règles. La première consiste à générer des motifs d'arbres, ou sous-arbres, contenant des contraintes sur les valeurs des attributs aux nœuds. L'objectif est de faire émerger des relations décisives entre les nœuds lors de l'induction. Nous avons défini un opérateur de spécialisation de tels motifs afin de mettre en œuvre, au sein du système de Programmation Logique Inductive (PLI) Aleph, une recherche descendante dans l'espace des hypothèses. La deuxième approche repose sur la définition experte d'attributs agré-

gats synthétisant l'information contenue dans les structures d'arbre, ceci afin d'utiliser un système d'induction en logique attribut-valeur, en l'occurrence CN2.

Dans notre application, l'approche PLI permet de conserver, au sein des motifs, la structure spatiale en arbres utilisée pour le modèle. Les motifs font ressortir les connexions importantes entre parcelles, on y voit apparaître essentiellement les données spatiales et topologiques. En comparaison avec la deuxième approche, la mise en œuvre a demandé un important travail de développement ; de plus, les temps d'apprentissage eux-mêmes sont beaucoup plus importants. Nous avons constaté également que la définition des attributs agrégats est un travail intéressant car ils permettent de dégager des variables pertinentes des sorties de simulation. D'autre part, cette approche permet de faire ressortir, plus clairement que dans l'approche PLI, les données de décision des agriculteurs.

Nous avons fait le choix d'induire un ensemble de règles important afin de proposer différentes explications des sorties de simulation. C'est pourquoi nous avons également proposé des méthodes d'analyse de règles. En particulier, nous souhaitons les utiliser pour suggérer des actions afin d'améliorer une situation. Nous avons alors défini une tâche originale de recommandation d'actions. L'utilisateur propose une situation qui lui est défavorable, les actions proposées doivent alors permettre d'améliorer la classe de la situation. Nous nous sommes intéressés aux notions de faisabilité et de qualité d'une action pour la recherche d'actions. Pour cette recherche, notre approche repose sur un ensemble de règles de classification exprimées dans une logique attribut-valeur et nous avons développé deux algorithmes. Le premier repose sur une recherche par faisceau des actions au cours de laquelle les actions sont spécialisées. Le second construit dans certaines conditions un optimum de l'espace des actions en combinant plusieurs règles.

Un ensemble de règles appris automatiquement au sein du projet a été utilisé afin de comparer les deux algorithmes pour la recherche d'actions. La comparaison a porté en particulier sur leur efficacité à produire des actions de qualité importante en fonction de la faisabilité des actions. On constate alors que si la faisabilité est sévère, c'est-à-dire que peu d'actions sont considérées possibles, alors le premier algorithme est plus efficace, sinon, c'est le contraire.

L'outil SACADEAU

Le projet SACADEAU a comme objectif final de fournir un outil permettant aux gestionnaires d'un territoire d'accompagner leurs réflexion et décisions sur les moyens de modifier leurs pratiques et d'améliorer les aménagements, ceci dans le but de maîtriser la qualité de l'eau. Les connaissances acquises au sein du projet interviennent à différents niveaux de l'outil SACADEAU (figure 5.1). Nous distinguons trois types de connaissances. Le travail effectué durant cette thèse porte sur chacune de ces parties.

Le premier type de connaissances disponibles est lié à la typologie des entrées. Cette typologie peut être utilisée pour la définition d'un langage de scénario, qui donne la possibilité à un acteur terrain de définir des configurations propres au problème qu'il considère, qu'il peut ensuite simuler à l'aide du modèle. Un premier usage de l'outil

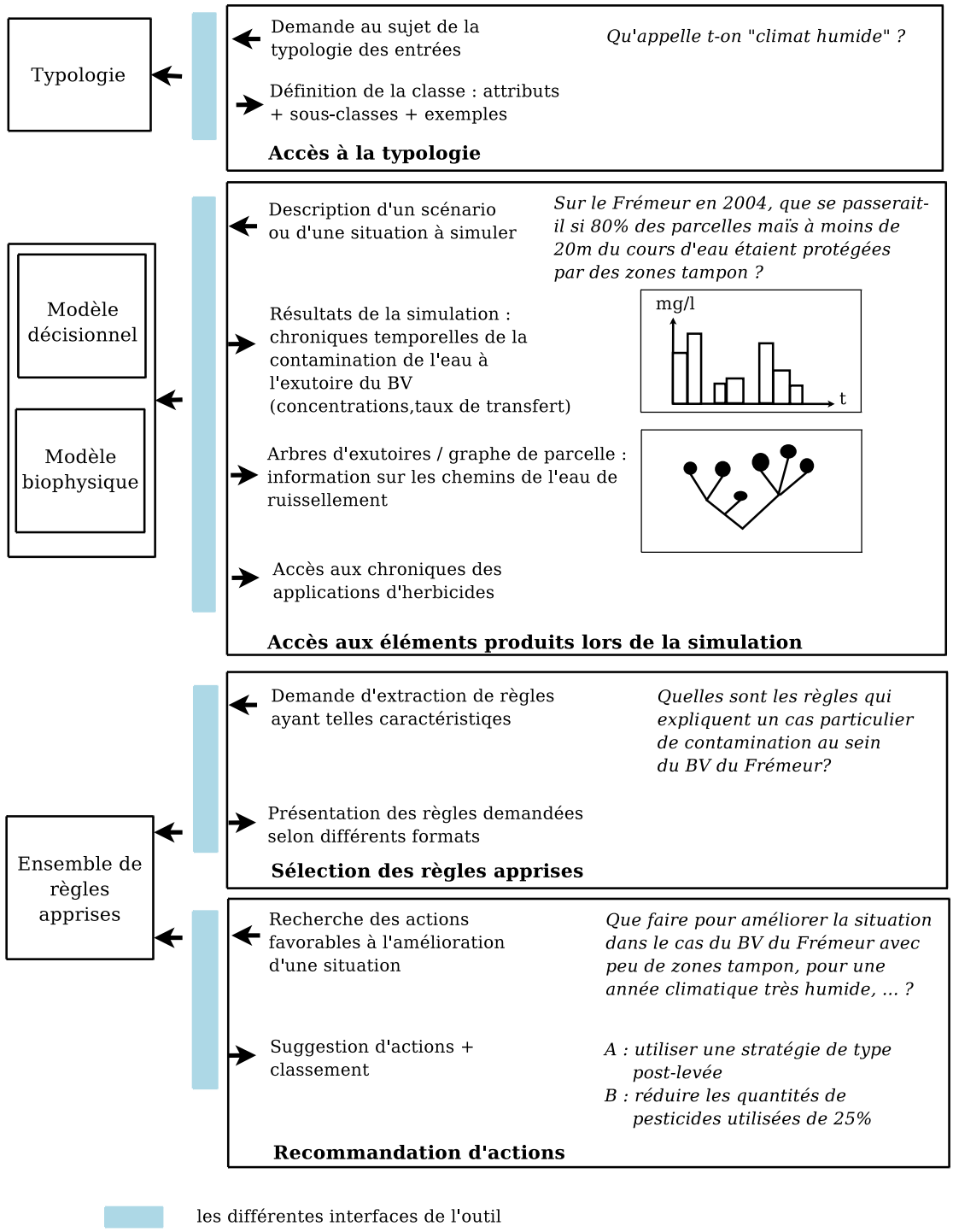


FIG. 5.1 – Différents modes d'usage de l'outil SACADEAU et ses différentes interfaces.

consiste à accéder à cette typologie afin de connaître précisément la définition des termes des différentes typologies. Il s'agit d'un accès au "dictionnaire" de l'application. Par exemple, une première étape du projet a permis la définition d'une typologie sur les climats. Si le langage de scénario n'est pas encore totalement défini, une base de scénarios est disponible, chacun ayant été développé dans le cadre d'études de l'impact des entrées sur les sorties du modèle.

Le second type de connaissances disponibles est lié au simulateur et à la modélisation. Le simulateur utilise les modèles décisionnel et biophysique et calcule les transferts de polluants pour une situation décrite par l'utilisateur, c'est-à-dire une configuration d'entrées. Le premier usage pour l'utilisateur est de visualiser les résultats de cette simulation. Il peut ainsi obtenir les quantités de pesticides arrivant à l'exutoire par ruissellement et écoulement de subsurface sous forme d'une chronique temporelle. Le second usage lié à la simulation ne concerne pas directement les résultats de la simulation mais la manière dont ils ont été obtenus. Pour les écoulements par ruissellement, la structure d'arbre d'exutoires est intéressante car elle permet de bien comprendre l'impact des itinéraires techniques de désherbage, de la topologie et des aménagements (fossés, haies, bandes enherbées) sur la connectivité des flux entre parcelles dans le bassin versant. L'utilisateur peut visualiser cette structure et en tirer des informations utiles, par exemple pour l'aménagement spatial du bassin. Il peut aussi consulter la chronique et la localisation des applications, résultats du modèle décisionnel.

Le troisième type de connaissances disponibles est lié aux connaissances apprises à partir des résultats des simulations, ceux-ci étant utilisés comme exemples d'apprentissage dans un processus d'induction automatique de règles de classification. Deux modes d'usage existent pour exploiter ce type de connaissances apprises. Le premier permet à l'utilisateur de visualiser les règles grâce à un langage de requêtes afin, par exemple, d'extraire celles qui, pour une situation particulière permettent d'expliquer son impact sur la qualité de l'eau. Le second usage utilise un outil dédié à la recommandation d'actions. Celui-ci, s'appuyant sur les règles, suggère des actions qui amélioreraient une situation jugée insatisfaisante par l'utilisateur, telle qu'une situation de contamination des eaux.

Perspectives

Un des premiers intérêts de l'intégration d'un modèle de simulation dans un système d'aide à la décision est la simulation de scénarios. Un scénario exprime une question du type "que se passerait-il si" ou plus précisément comment évoluent les sorties du modèle si les entrées varient d'une certaine façon. Dans ce travail de thèse, nous avons simulé trois scénarios qui ont été définis de manière à évaluer l'impact de différents facteurs d'entrées. Un scénario est mis en œuvre ici par un ensemble de contraintes sur les entrées. Plusieurs simulations, dont les entrées respectent ces contraintes, sont alors effectuées afin de produire une réponse globale au scénario. Un travail intéressant serait d'automatiser le processus de traduction d'un scénario, exprimé dans un langage haut niveau, en un ensemble de contraintes. Une possibilité pour cela est de définir le langage

de scénarios, celui-ci s'appuyant sur les typologies définies pour les entrées. Il serait possible de résumer les résultats de scénario en utilisant des techniques d'apprentissage de règles semblables à celles utilisées dans cette thèse. Pour un tel apprentissage, la simulation de différents scénarios peut ainsi permettre la génération de différentes bases d'apprentissage.

Le modèle de simulation considère plusieurs granularités spatiales. La parcelle est l'unité fonctionnelle des pratiques agricoles, certaines cependant sont communes aux parcelles qui forment une exploitation. L'exutoire de parcelle est l'unité fonctionnelle du transfert d'eau et pesticides. Un arbre d'exutoires est une partie de bassin versant, composés d'exutoires de parcelles, indépendante vis-à-vis du transfert. Pour l'apprentissage de règles, nous avons travaillé à l'échelle de l'arbre d'exutoires. Techniquement, pour exprimer cette structure, un système de PLI a été utilisé mais d'autres possibilités existent. Par exemple, Ferré et King (Ferré & King, 2005) proposent d'utiliser des logiques dédiées. Celles-ci permettent la recherche de motifs d'arbre d'exutoires si on utilise un foncteur pour les arbres et un foncteur pour les attributs aux nœuds de l'arbre ; ces foncteurs doivent préalablement être définis. Bien que l'arbre d'exutoires est l'échelle qui nous a paru pertinente, d'autres échelles auraient pu être considérées telles que l'exploitation, qui centralise des décisions liées aux pratiques agricoles.

Nous nous sommes intéressés uniquement aux relations spatiales, mais il existe des relations temporelles que l'on sait a priori importantes. Ainsi, le délai entre une application de pesticides et une première pluie importante est un facteur déterminant pour le transfert. Plus généralement, les choix d'échelle et ceux de la gestion de données spatiales ou temporelles est une problématique intéressante. Ainsi, si l'on s'intéresse au problème du transfert des nitrates, l'infiltration dans le sol est un transfert plus important que celui du ruissellement et les délais de transfert vers le réseau hydrographique sont plus longs. Les échelles spatiales et temporelles à considérer doivent donc être différentes.

La tâche de recommandation d'actions telle que définie en 4.24, consiste en la recherche d'actions valides (définition 4.22) et donc nous nous restreignons à des actions qui sont des corrections (définition 4.16). Si on considère aussi les actions qui sont des précisions, cela nous permet de redéfinir la tâche pour la recherche de règles actions (Ras & Tsay, 2003). En effet, dans ce cas, on ne considère pas seulement une situation défavorable mais l'ensemble des situations couvertes par un complexe qui n'est pas totalement instancié.

Plus généralement, les différentes problématiques du projet SACADEAU sont également des problématiques du projet APPEAU. Ce dernier a pour objectif de contribuer à améliorer les prises de décision collective dans le cadre de la gestion de la ressource en eau à l'échelle d'un territoire (bassin versant, périmètre irrigué ...). En effet, la raréfaction et la dégradation de la qualité des ressources en eau pose la question de la durabilité des usages actuels de cette ressource. Parmi ceux-ci, l'agriculture occupe une place prépondérante du fait des montants qu'elle prélève et de la pollution qu'elle génère. Le projet APPEAU vise plus spécifiquement à développer des outils et méthodes à base de modèles mathématiques permettant l'évaluation de scénarios en vue d'une meilleure planification des activités agricoles et des ressources en eau. Les approches étudiées dans cette thèse seront utilisées et étendues dans le cadre de ce nouveau projet.

Annexe A

La logique des prédicats et Prolog

Cette annexe présente la logique des prédicats, souvent assimilée à la logique du premier ordre. Elle présente également le langage Prolog, qui s'appuie sur cette logique, et son fonctionnement.

A.1 Le langage des prédicats

Le vocabulaire de la logique des prédicats comprend :
- des variables
- des constantes
- des fonctions
- des prédicats
- des connecteurs : $\rightarrow$ (implication), $\wedge$ (et), $\vee$ (ou) et $\neg$ (non)
- des quantificateurs de variable : $\forall$ (quantificateur universel) et $\exists$ (quantificateur existentiel).

Par convention, un identifiant dont la première lettre est en majuscule représente une variable. L'arité d'une fonction, ou d'un prédicat, est son nombre d'arguments. Par exemple le prédicat p d'arité 2 est notée $p/2$. Un prédicat diffère d'une fonction du fait qu'il est toujours à valeur dans $\{vrai, faux\}$.

Définition A.1 (Terme).
Un terme est défini récursivement comme étant :
- *soit une constante*
- *soit une variable*
- *soit une fonction appliquée à des termes, par exemple $f(t_1, t_2)$ où t_1 et t_2 sont des termes.*

Définition A.2 (Littéral).
Un littéral est un prédicat appliqué à des termes et éventuellement précédé du connecteur $\neg$. Par exemple, $p(t_1, t_2, t_3)$, où p est un prédicat d'arité 3 et t_1, t_2, t_3 sont des termes, est un littéral positif et $\neg p(t_1, t_2, t_3)$ est un littéral négatif.

Définition A.3 (Formule).
Une formule est définie récursivement :
- *un littéral est une formule*
- *si α et β sont deux formules, alors $\alpha \rightarrow \beta$, $\neg\alpha$, $\neg\beta$, $\alpha \wedge \beta$, $\alpha \vee \beta$ sont des formules*
- *si X est une variable et α une formule alors $(\forall X)\alpha$ et $(\exists X)\alpha$ sont des formules.*

Un terme, un littéral et une formule ne contenant aucune variable sont dits *instanciés*. Une formule est dite *bien formée* si toutes ses variables sont quantifiées une fois et une seule (voir l'exemple A.15).

Définition A.4 (Clause).
Une clause est une formule composée d'une disjonction finie de littéraux dans laquelle toutes les variables sont quantifiées universellement.

Définition A.5 (Forme clausale).
Une formule sous forme clausale est une conjonction de clauses.

Une formule peut être traduite en forme clausale qui est donc une formule s'appuyant sur les seuls connecteurs $\neg$, $\wedge$ et $\vee$ et où les variables sont toutes universellement quantifiées. On obtient la forme clausale β d'une formule α en procédant en trois étapes (voir l'exemple A.15) :

1. mise sous forme prénexe de α : séparation de α en une suite de quantificateurs et une formule (appelée matrice) sans quantificateur
2. mise sous forme normale de Skolem : élimination des quantificateurs existentiels, les variables quantifiées existentiellement deviennent de nouvelles fonctions des variables quantifiées universellement
3. mise sous forme clausale de la matrice.

A.2 Satisfaisabilité d'un ensemble de formules

Définition A.6 (Interprétation).
Une interprétation d'une formule bien formée consiste à :
- *définir un domaine d'interprétation $D \neq \emptyset$*
- *assigner à chaque constante un élément de D*
- *pour chaque fonction d'arité i, assigner une application $D^i \rightarrow D$*
- *pour chaque prédicat d'arité i, assigner une application $D^i \rightarrow \{vrai, faux\}$*

La valeur d'une formule selon une interprétation prend sa valeur dans $\{vrai, faux\}$.

Notons I une interprétation et D son domaine. Pour calculer la valeur d'une formule selon I, on remplace toutes les constantes puis toutes les fonctions par leur valeur assignée dans D. Ensuite on remplace tous les prédicats par leur valeur dans $\{vrai, faux\}$ et on évalue la valeur dans $\{vrai, faux\}$ de la formule à l'aide des connecteurs logiques (par exemple $vrai \wedge vrai$ donne $vrai$).

Notons α une formule contenant la variable X. la valeur d'une formule $(\forall X)\alpha$ selon I est *vrai* si, pour toutes les valeurs d de D, la valeur de la formule α, où X a été remplacée par d, est *vrai*. La valeur de la formule $(\exists X)\alpha$ selon I est *vrai* s'il existe au moins une valeur d de D telle que la valeur de la formule α, où X a été remplacée par d, est *vrai*.

Définition A.7 (Modèle).
Une interprétation I est un modèle d'une formule α si la valeur de α selon I est vrai. De plus, I est un modèle d'un ensemble de formules $\mathcal{H}$ si elle est un modèle de toute formule de $\mathcal{H}$.

Définition A.8 (Satisfaisabilité).
Un ensemble de formules $\mathcal{H}$ est satisfaisable si et seulement si il existe un modèle de $\mathcal{H}$.

Soit α une formule et β sa forme clausale alors α est satisfaisable si et seulement si β est satisfaisable.

Définition A.9 (Conséquence logique).
Une formule α est conséquence logique d'un ensemble de formules $\mathcal{H}$ si et seulement si tout modèle de $\mathcal{H}$ est un modèle de α. On note alors $\mathcal{H} \models \alpha$.

Il est nécessaire pour vérifier la conséquence logique entre l'ensemble de formules $\mathcal{H}$ et la formule α de calculer tous les modèles de $\mathcal{H}$, or le nombre d'interprétations, et donc de modèles potentiels, est infini. En réalité, on s'intéresse au sous ensemble des interprétations de Herbrand.

Définition A.10 (Univers de Herbrand).
L'univers de Herbrand d'un ensemble de formules $\mathcal{H}$ est l'ensemble de tous les termes instanciés pouvant être formé à partir des symboles de fonctions et constantes utilisées dans $\mathcal{H}$.

Définition A.11 (Base de Herbrand).
La base de Herbrand d'un ensemble de formules $\mathcal{H}$ est l'ensemble de tous les littéraux positifs instanciés qui peuvent être formés à partir des symboles de prédicats utilisés dans $\mathcal{H}$ et les termes instanciés de l'univers de Herbrand.

Définition A.12 (Interprétation de Herbrand).
Une interprétation d'Herbrand est un sous ensemble de la base de Herbrand, elle assigne la valeur vrai à tout élément de ce sous ensemble et $faux$ à tout autre élément de la base.

Sous certaines conditions, les interprétations d'Herbrand suffisent à montrer la relation de conséquence logique.

Définition A.13 (Clause définie).
Une clause est définie si elle possède exactement un littéral positif.

Définition A.14 (Programme défini).
Un programme défini est une conjonction de clauses définies, c'est-à-dire une forme clausale où toutes les clauses sont définies.

Si un programme défini $\mathcal{H}$ est satisfaisable alors il possède un unique plus petit modèle de Herbrand, c'est-à-dire une interprétation de Herbrand de taille minimale. C'est celui qui est utilisé pour prouver la satisfaisabilité de $\mathcal{H}$ ou son insatisfaisabilité. Intuitivement, il s'agit du plus petit ensemble de connaissances, s'appuyant sur les constantes du programme, qui permet de satisfaire $\mathcal{H}$ (voir l'exemple A.17).

Exemple A.15 (formules bien formées et mise sous forme clausale).
Voici deux exemples de formules mal formées ; dans la première, la variable X n'est pas quantifiée, dans la seconde, X est quantifiée deux fois :

$$p(X) \wedge p(f(a))$$

$$(\forall X p(X)) \wedge (\exists X q(X))$$

Une formule bien formée F :

$$\forall X((\forall Y(p(Y) \vee q(a, X, Y))) \rightarrow r(X))$$

Mise sous forme prénexe de F (étape 1 de la mise sous forme clausale) :

$$\forall X(\neg(\forall Y(p(Y) \vee q(a, X, Y))) \vee r(X))$$

$$\forall X((\exists Y(\neg p(Y) \wedge \neg q(a, X, Y))) \vee r(X))$$

$$\forall X \exists Y((\neg p(Y) \wedge \neg q(a, X, Y)) \vee r(X))$$

Mise sous forme normale de Skolem (étape 2 de la mise sous forme clausale) :

$$\forall X((\neg p(f(X)) \wedge \neg q(a, X, f(X))) \vee r(X))$$

Mise sous forme clausale de F :

$$\forall X((\neg p(f(X)) \vee r(X)) \wedge (\neg q(a, X, f(X)) \vee r(X)))$$

Le programme défini, dans le langage Prolog, correspondant à la forme clausale (cette étape est possible puisque les clauses de la forme clausale sont définies) :

```
r(X) :- p(f(X)).
r(X) :- q(a,X,f(X)).
```

Le symbole de fonction f qui pour tout X donne le terme qui rend vrai les deux clauses précédentes. En pratique dans Prolog, la variable Y est conservée, l'étape de mise sous forme de Skolem n'est pas appliquée et le programme Prolog, où Y est quantifiée existentiellement, s'écrit :

```
r(X) :- p(Y).
r(X) :- q(a,X,Y).
```

A.3 Le langage Prolog

Un programme Prolog est un programme défini avec une notation différente. Par exemple, la formule $\forall X \forall Z \forall Y (p(X,Z) \vee \neg p(X,Y) \vee \neg p(Y,Z))$ est une forme clausale de la formule $\forall X \forall Z \exists Y (p(X,Z) \leftarrow (p(X,Y) \wedge p(Y,Z)))$. En Prolog le programme défini est constitué d'une seule clause `p(X,Z):-p(X,Y),p(Y,Z)`. La tête de clause Prolog est `p(X,Z)` et le corps `p(X,Y),p(X,Z)`. Les quantifieurs de variables ne sont pas apparents, il s'agit de quantifieurs en début de clause, universels pour les variables de la tête de clause (ici X et Z) et existentiels sinon (ici Y). Un autre exemple de traduction de formule en programmes Prolog est donné dans l'exemple A.15.

Définition A.16 (Substitution).
Une substitution (notée θ) est une fonction de l'ensemble des variables dans l'ensemble des termes ou variables. Elle est souvent proposée sous une forme de liste de couples, par exemple $\theta = [X_1/t_1, \ldots, X_n/t_n]$. A X_i, la substitution θ associe t_i. Une substitution θ s'applique à une formule α en y substituant chacune des variables par le terme associé, la formule résultante est notée $\alpha\theta$.

Une substitution θ est un *unificateur* de deux littéraux l_1 et l_2 si $l_1\theta = l_2\theta$. Elle est le *plus grand unificateur* de l_1 et l_2 si, pour toute autre substitution θ_1 unificateur de l_1 et l_2, $\theta \subset \theta_1$.

Le mécanisme de preuve de but[1] dans Prolog est la *résolution SLD*. Le résultat d'une preuve d'un but B est si il est démontrable, une substitution entre les variables de B et des constantes du programme. Prolog construit une preuve récursivement de la manière suivante (voir l'exemple A.17) :

- Pour une clause C du programme dont la tête T est unifiable avec le fait B, le plus grand unificateur θ entre T et B est construit.
- Si le corps de C est vide alors la preuve est θ. Sinon, si $l_1, l_2, \ldots, l_n$ est le corps de C alors une preuve θ_1 de $l_1\theta$ est construite, puis une preuve θ_2 de $l_2(\theta \cup \theta_1)$, et ceci jusque n. La preuve de B est alors $\theta \cup \theta_1 \cup \ldots \cup \theta_n$.

Lors de la démonstration, il est possible qu'un littéral ne soit unifiable avec aucune tête de clause, le but est alors indémontrable et donc considéré faux, c'est le principe de la *négation par échec*. D'un autre côté, si un littéral est unifiable avec plusieurs têtes de clauses, alors cela peut amener à construire plusieurs preuves.

Exemple A.17 (modèle de Herbrand et déduction Prolog).
Soit le programme Prolog constitué des clauses numérotées suivantes :
(1) `p(a).`
(2) `p(b).`
(3) `q(b).`
(4) `r(X) :-p(X),q(X).`
La base de Herbrand est l'ensemble des littéraux $\{p(a), p(b), q(a), q(b), r(a), r(b)\}$. Le plus petit modèle de Herbrand de ce programme défini est $\{p(a), p(b), q(b), r(b)\}$.

[1] un but Prolog est un littéral positif, c'est-à-dire une clause définie sans littéral négatif

La figure A.18 montre la construction d'une preuve (c'est-à-dire une substitution) du but `r(X)` . Schématiquement, Prolog maintient une pile d'exécution composée des buts qu'il reste à prouver, il effectue les substitutions résultant des preuves précédentes pour mettre à jour la pile. S'il tombe sur un échec, le mécanisme de *backtracking* permet d'essayer d'autres clauses.

Ici, on empile tout d'abord le but `r(X)` . Pour le prouver il faut prouver `p(X)` puis `q(X)` (clause 4), ces deux buts remplacent alors le premier. Pour prouver `p(X)` , une première solution est de substituer `X` à `a` mais le système ne peut pas prouver `q(a)` . Le système essaye alors (après un *backtrack*) de prouver `p(X)` en substituant `X` à `b` . Cette fois ci, la preuve de `q(b)` existe. Le résultat est une substitution de `X` par `b` .

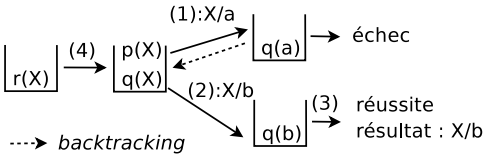


FIG. A.18 – Exemple de déduction (ou preuve de but) Prolog.

Annexe B

Les attributs d'exutoire et agrégats

Cette annexe détaille les différents attributs d'exutoires (voir définition 3.14) et les attributs agrégats (voir définition 3.22). On s'intéresse ici au calcul des valeurs de ces attributs, à leur domaine et à leur interprétation.

B.1 Les attributs d'exutoires

Les attributs d'exutoires sont utilisés dans l'apprentissage de motifs d'arbre d'exutoires (voir section 3.4), la liste de ces attributs est donnée dans le tableau 3.15. Pour un exutoire donné, les attributs suivants sont définis :

- une valeur de l'attribut quantitatif $os\_surface$ représente la surface (en hectares) de l'exutoire, il varie pour un exutoire entre 0,04 ha et 7,88 hectares.
- une valeur de l'attribut os_mais est un booléen à $vrai$ si l'exutoire est un exutoire de parcelle cultivée en maïs.
- une valeur de l'attribut $os\_dispositif\_tampon$ est un booléen à vrai si l'exutoire est équipé d'un dispositif tampon tel qu'une bande enherbée.
- l'attribut os_pente est un attribut qualitatif représentant la pente locale à l'exutoire. Elle prend ses valeurs dans $\{nulle, faible, forte\}$. Une pente $nulle$ est une pente locale inférieure ou égale à 0.215%, une pente $faible$ est une pente supérieure à 0.215% et inférieure ou égale à 3.119% et une pente forte est une pente supérieure à 3.119%.
- une valeur de l'attribut quantitatif os_mrt représente la valeur d'un indice topographique variant entre -2,083 et 386,750. Cet indice est un indice de similarité hydrologique utilisé dans le modèle TOPMODEL (Beven & Kirkby, 1979). Il permet d'estimer le déficit local en eau du sol par rapport à un déficit moyen calculé pour le bassin versant. Un déficit en eau définit la quantité d'eau qu'il faudrait infiltrer pour saturer un profil de sol. Plus l'indice topographique d'un exutoire est important, moins la quantité d'eau qu'il faudrait pour saturer les sol est important et plus la nappe est proche du sol. Les exutoires à l'indice topographique important sont en majorité des exutoires de bas de bassin versant.
- l'attribut itk_type prend ces valeurs dans $\{tout\_en\_post, pre\_puis\_post\}$. Un

ITK de type *pre_puis_post* consiste en une application dès le semis du maïs et un autre au stade 5 feuilles des épis de maïs. Un ITK de type *tout_en_post* consiste en une application de pesticides autour du stade 3 feuilles et une autre application entre les stades 5 feuilles et 7 feuilles du maïs. Le type d'ITK définit donc les fenêtres temporelles des applications, les dates exactes elles sont simulées à l'aide du modèle décisionnel. En fait des règles expertes permettent de prendre en compte des contraintes de faisabilité du traitement au jour le jour, contraintes qui sont liées par exemple à la pluie journalière ou la disponibilité des machines de traitement.

- l'attribut quantitatif *itk_pression* représente la quantité totale d'herbicides appliquée sur l'exutoire, toute matière active confondue. Elle varie entre 0 et 24 880 grammes.

B.2 Les attributs agrégats

Les attributs agrégats sont utilisés dans l'apprentissage de règles attribut-valeur (voir section 3.5), la liste de ces attributs est donnée dans le tableau 3.23. Les domaines des attributs agrégats quantitatifs sont donnés pour un arbre d'exutoire, et donc calculé au niveau de l'exutoire racine des arbres :

- L'attribut quantitatif *agr_pression* représente la quantité totale d'herbicides appliquée sur les exutoires de l'arbre, toute matière active confondue. Il varie entre 6 et 54 645 grammes.
- L'attribut quantitatif $agr\_rapport\_risque\_faible$ représente le pourcentage de de matière active, du groupe risque faible, appliquée [1]. Il varie entre 4 et 100 %.
- L'attribut quantitatif $agr\_rapport\_risque\_fort$ représente le pourcentage de de matière active, du groupe risque fort, appliquée. Il varie entre 0 et 96%.
- L'attribut quantitatif *agr_rapport_prelevee* représente le pourcentage totale de pesticides appliquée au stade de prélevée, c'est-à-dire au moment du semis. Il varie entre 0 et 96%.
- L'attribut quantitatif $agr\_surface$ représente la surface totale de l'arbre d'exutoires Il varie entre 800 et 364 800 mètres carrés.
- L'attribut quantitatif $agr\_rapport\_surf\_mais$ représente le pourcentage de surface maïs sur l'ensemble de l'arbre d'exutoires. Il varie entre 0 et 100%.
- L'attribut quantitatif $agr\_surf\_max\_mais$ représente la surface de l'exutoire de parcelle cultivée en maïs le plus important de l'arbre. Il varie entre 400 et 66 800 mètres carrés.
- L'attribut quantitatif *agr_rapport_disp_tampon* représente le pourcentage de surface de l'arbre d'exutoires utilisée en dispositif tampon. Il varie entre 0 et 67%.
- L'attribut quantitatif $agr\_max\_profondeur$ représente la profondeur maximale de l'arbre (en nombre d'exutoires). Il varie entre 1 et 24.

[1] La CORPEP (http ://draf.bretagne.agriculture.gouv.fr/corpep/)propose en effet une classification des matières actives en trois groupes en fonction des quantités usuelles appliquées, leur temps de demi-vie et leur constante de rétention.

- L'attribut qualitatif $agr\_topo\_forme\_denivele$ prend une valeur parmi *concave*, *convexe*, *pentue* et *plate*. Une forme d'arbre d'exutoires *concave* signifie qu'il existe des pentes locales importantes en haut de l'arbre d'exutoires (c'est-à-dire sur les exutoires amont) et des pentes faibles sur les exutoires du bas. Une forme *convexe* signifie, au contraire, que les pentes importante se trouvent en aval dans l'arbre et les pentes faibles en amont. Un arbre d'exutoires de forme *pentue* possède des pentes importantes sur l'ensemble des exutoires, et au contraire les exutoires d'un arbre d'exutoires de forme *plate* ont des pentes plutôt faibles. Pour calculer cet attribut, nous répartissons les exutoires d'un arbre dans deux groupes de taille égale, les exutoires amont et les exutoires aval. Nous calculons les pentes moyennes pour chacun des deux groupes et nous comparons ces pentes moyennes au seuil de 3%. Si la pente moyenne des exutoires amont est supérieure à 3% et la pente moyenne des exutoires aval est inférieure à 3% alors la forme de l'arbre est *concave*, etc...
- L'attribut qualitatif $agr\_topo\_forme\_largeur$ prend ses valeurs dans l'ensemble $\{u, v, i, isole\}$. Un arbre d'exutoires de forme u possédant de nombreux exutoires en aval, ce qui implique un nombre important d'exutoires proches du réseau hydrographique. Au contraire, un arbre de forme i est un arbre linéaire en aval. Un arbre de forme $isole$ signifie que l'arbre ne possède qu'un seul exutoire. Pour le calcul de cet attribut, nous nous intéressons au nombre d'exutoires $nb\_exus\_prof(p)$ à une profondeur donnée p de l'arbre. Si $nb\_exus\_prof(1) = 0$ alors la forme de l'arbre est $isole$, sinon, si $nb\_exu\_prof(p) = 1$ pour tout p sa forme est i, sinon, si pour une profondeur $p < 3$, $nb\_exu\_prof(p) > 2$ alors sa forme est u, sinon sa forme est v.

Annexe C

Les règles apprises

Les ensembles des règles générées à partir des apprentissages décrits dans le chapitre 3 sont donnés ici.

C.1 Découverte de motifs d'arbres d'exutoires

Nous donnons ici l'ensemble des 28 motifs d'arbres d'exutoires résultant de l'apprentissage exposé dans la section 3.4.

```
[theorie transfert_important]
[Rule 1] [Pos cover = 104 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mais,D), eq(D,true),
 intro_noeud(A,C,[C,B],E), value(A,E,os_surface,F), gteq(F,0.32),
 intro_noeud(A,C,[E,C,B],G),
 intro_noeud(A,B,[G,E,C,B],H), value(A,H,itk_pression,I), gteq(I,17.0).
[Rule 2] [Pos cover = 74 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), value(A,B,itk_pression,C), lteq(C,2384.0), value(A,B,os_surface,D),
 gteq(D,0.28).
[Rule 3] [Pos cover = 51 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D),
 value(A,D,itk_pression,E), gteq(E,502.0), intro_noeud(A,D,[D,C,B],F),
 intro_noeud(A,F,[F,D,C,B],G), value(A,G,os_surface,H), gteq(H,0.24).
[Rule 4] [Pos cover = 57 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,B,[C,B],D), value(A,D,os_mrt,E),
 gteq(E,36.25), value(A,D,itk_pression,F), lteq(F,3650.0).
[Rule 5] [Pos cover = 46 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mrt,D), gteq(D,36.25),
 intro_noeud(A,C,[C,B],E), intro_noeud(A,C,[E,C,B],F), intro_noeud(A,C,[F,E,C,B],G),
 intro_noeud(A,B,[G,F,E,C,B],H), intro_noeud(A,H,[H,G,F,E,C,B],I),
 value(A,I,os_mais,J), eq(J,true).
[Rule 6] [Pos cover = 61 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mrt,D), gteq(D,51.25),
```

```
 intro_noeud(A,C,[C,B],E), intro_noeud(A,E,[E,C,B],F), intro_noeud(A,F,[F,E,C,B],G),
 intro_noeud(A,G,[G,F,E,C,B],H), intro_noeud(A,F,[H,G,F,E,C,B],I),
 value(A,I,itk_pression,J), gteq(J,27.0).
[Rule 7] [Pos cover = 56 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_surface,D), lteq(D,0.84),
 intro_noeud(A,C,[C,B],E), intro_noeud(A,C,[E,C,B],F), intro_noeud(A,C,[F,E,C,B],G),
 value(A,G,os_mrt,H), gteq(H,8.875), value(A,G,os_mais,I), eq(I,true).
[Rule 8] [Pos cover = 109 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mais,D), eq(D,true),
 value(A,C,os_surface,E), gteq(E,0.96), value(A,C,os_dispositif_tampon,F), eq(F,false).
[Rule 9] [Pos cover = 30 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,itk_pression,D), lteq(D,181.0),
 value(A,C,os_mrt,E), gteq(E,7.25), intro_noeud(A,C,[C,B],F), value(A,F,os_mais,G),
 eq(G,false).
[Rule 10] [Pos cover = 52 Neg cover = 1]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D), value(A,D,os_pente,E),
 eq(E,nulle), intro_noeud(A,D,[D,C,B],F), intro_noeud(A,D,[F,D,C,B],G),
 intro_noeud(A,D,[G,F,D,C,B],H), intro_noeud(A,D,[H,G,F,D,C,B],I),
 value(A,I,os_surface,J), gteq(J,0.48), value(A,I,os_mrt,K), gteq(K,30.5).
[Rule 11] [Pos cover = 152 Neg cover = 1]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_pente,D), eq(D,nulle),
 value(A,C,itk_pression,E), lteq(E,3903.0), intro_noeud(A,C,[C,B],F),
 value(A,F,os_surface,G), gteq(G,0.08).
[Rule 12] [Pos cover = 41 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mais,D), eq(D,true),
 value(A,C,os_mrt,E), gteq(E,16.75), value(A,C,os_surface,F), lteq(F,0.68).
[Rule 13] [Pos cover = 60 Neg cover = 2]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D),
 intro_noeud(A,B,[D,C,B],E), value(A,E,os_pente,F), eq(F,nulle),
 intro_noeud(A,B,[E,D,C,B],G), intro_noeud(A,B,[G,E,D,C,B],H),
 intro_noeud(A,B,[H,G,E,D,C,B],I), value(A,I,itk_pression,J),
 lteq(J,1379.0), value(A,I,os_mrt,K),
 gteq(K,3.5), value(A,I,os_surface,L), lteq(L,0.52).
[Rule 14] [Pos cover = 70 Neg cover = 6]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D), value(A,D,os_mais,E),
 eq(E,true), intro_noeud(A,D,[D,C,B],F), value(A,F,itk_pression,G), gteq(G,169.0).

[theorie transfert_faible]
[Rule 1] [Pos cover = 24 Neg cover = 0]
transfert_faible(A) :-
 root(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D),
 value(A,D,os_dispositif_tampon,E), eq(E,false), intro_noeud(A,B,[D,C,B],F),
 intro_noeud(A,B,[F,D,C,B],G), intro_noeud(A,B,[G,F,D,C,B],H),
 intro_noeud(A,B,[H,G,F,D,C,B],I), value(A,I,os_mrt,J), lteq(J,3.125),
 value(A,I,os_pente,K), eq(K,forte), value(A,I,os_surface,L), gteq(L,0.24),
 value(A,I,os_mais,M), eq(M,false).
```

```
[Rule 2] [Pos cover = 15 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_dispositif_tampon,D), eq(D,true),
 intro_noeud(A,B,[C,B],E), value(A,E,os_pente,F), eq(F,forte).
[Rule 3] [Pos cover = 46 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_surface,D), lteq(D,0.52),
 value(A,C,os_mrt,E), lteq(E,3.125), intro_noeud(A,C,[C,B],F),
 intro_noeud(A,F,[F,C,B],G), intro_noeud(A,G,[G,F,C,B],H),
 intro_noeud(A,H,[H,G,F,C,B],I), value(A,I,os_pente,J), eq(J,forte).
[Rule 4] [Pos cover = 33 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mrt,D), lteq(D,13.25),
 value(A,C,os_surface,E), gteq(E,0.68), intro_noeud(A,C,[C,B],F),
 intro_noeud(A,C,[F,C,B],G), intro_noeud(A,C,[G,F,C,B],H), value(A,H,os_pente,I),
 eq(I,forte).
[Rule 5] [Pos cover = 46 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mrt,D), lteq(D,3.125),
 value(A,C,os_pente,E), eq(E,forte), intro_noeud(A,C,[C,B],F),
 intro_noeud(A,C,[F,C,B],G), value(A,G,os_mais,H), eq(H,true).
[Rule 6] [Pos cover = 19 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,itk_pression,D), lteq(D,60.0),
 value(A,C,os_mrt,E), lteq(E,-0.25), intro_noeud(A,B,[C,B],F).
[Rule 7] [Pos cover = 34 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D),
 intro_noeud(A,D,[D,C,B],E), value(A,E,os_mrt,F), lteq(F,0.0),
 intro_noeud(A,E,[E,D,C,B],G), value(A,G,os_surface,H), gteq(H,0.08).
[Rule 8] [Pos cover = 23 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mrt,D), lteq(D,0.875),
 intro_noeud(A,C,[C,B],E), intro_noeud(A,E,[E,C,B],F), intro_noeud(A,B,[F,E,C,B],G),
 intro_noeud(A,G,[G,F,E,C,B],H), value(A,H,os_surface,I), gteq(I,0.24).
[Rule 9] [Pos cover = 17 Neg cover = 2]
transfert_faible(A) :-
 racine(A,B), value(A,B,os_mrt,C), lteq(C,6.25), value(A,B,os_pente,D),
 eq(D,nulle).
[Rule 10] [Pos cover = 18 Neg cover = 3]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mrt,D), lteq(D,0.75),
 value(A,C,itk_pression,E), gteq(E,100.0), value(A,C,os_pente,F), eq(F,forte).
[Rule 11] [Pos cover = 55 Neg cover = 5]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D), value(A,D,os_mrt,E),
 lteq(E,1.916667), intro_noeud(A,D,[D,C,B],F), intro_noeud(A,F,[F,D,C,B],G),
 intro_noeud(A,G,[G,F,D,C,B],H).
[Rule 12] [Pos cover = 66 Neg cover = 5]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D), value(A,D,os_mrt,E),
 lteq(E,0.375), value(A,D,os_surface,F), gteq(F,0.16).
[Rule 13] [Pos cover = 32 Neg cover = 8]
transfert_faible(A) :-
```

```
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mrt,D), lteq(D,-0.625),
 value(A,C,itk_pression,E), lteq(E,59.0).
[Rule 14] [Pos cover = 56 Neg cover = 11]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D),
 intro_noeud(A,D,[D,C,B],E), value(A,E,os_mrt,F), lteq(F,0.0),
 intro_noeud(A,E,[E,D,C,B],G), intro_noeud(A,B,[G,E,D,C,B],H),
 value(A,H,os_surface,I), lteq(I,0.32).
```

C.2 Découverte de règles attributs valeur

Nous donnons ici l'ensemble des 33 motifs d'arbres d'exutoires résultant de l'apprentissage exposé dans la section 3.5.

```
[theorie transfert_important]
IF    agr_rapport_risque_fort > 0.50
  AND agr_surf_max_mais > 15400.00
  THEN obj_class = transfert_important  [Pos cover=125 Neg cover=0]
IF    agr_pression > 218.50
  AND agr_surface > 4600.00
  AND agr_rapport_surf_mais > 71.50
  AND agr_surf_max_mais > 3800.00
  AND agr_rapport_disp_tampon < 26.50
  AND agr_max_profondeur < 6.50
  THEN obj_class = transfert_important  [Pos cover=105 Neg cover=0]
IF    84000.00 < agr_surface < 121800.00
  AND agr_surf_max_mais < 19800.00
  AND agr_max_profondeur > 5.50
  AND agr_topo_forme_largeur = u
  THEN obj_class = transfert_important  [Pos cover=39 Neg cover=0]
IF    agr_surface < 73600.00
  AND agr_rapport_surf_mais < 72.00
  AND agr_surf_max_mais > 11000.00
  THEN obj_class = transfert_important  [Pos cover=49 Neg cover=0]
IF    agr_surface > 164800.00
  AND agr_rapport_surf_mais > 10.00
  AND agr_surf_max_mais > 6800.00
  THEN obj_class = transfert_important  [Pos cover=91 Neg cover=1]
IF    agr_pression > 61.50
  AND agr_rapport_surf_mais > 53.50
  AND agr_rapport_disp_tampon < 16.50
  AND agr_topo_forme_denivele = plate
  THEN obj_class = transfert_important  [Pos cover=92 Neg cover=0]
IF    agr_surface < 159400.00
  AND agr_surf_max_mais > 9800.00
  AND 0.50 < agr_rapport_disp_tampon < 5.50
  THEN obj_class = transfert_important  [Pos cover=44 Neg cover=0]
IF    agr_surface < 76000.00
  AND agr_rapport_surf_mais < 54.00
  AND 3800.00 < agr_surf_max_mais < 8200.00
  AND agr_topo_forme_largeur = v
  THEN obj_class = transfert_important  [Pos cover=22 Neg cover=0]
IF    agr_rapport_risque_fort > 10.00
```

```
  AND agr_surface > 37800.00
  AND agr_rapport_surf_mais > 12.00
  AND agr_surf_max_mais < 10600.00
  AND agr_max_profondeur < 4.50
  THEN obj_class = transfert_important  [Pos cover=22 Neg cover=0]
IF    agr_rapport_surf_mais > 23.00
  AND agr_max_profondeur < 2.50
  AND agr_topo_forme_denivele = concave
  THEN obj_class = transfert_important  [Pos cover=16 Neg cover=0]
IF    agr_surface < 42400.00
  AND agr_surf_max_mais < 8800.00
  AND agr_max_profondeur > 7.50
  AND agr_topo_forme_largeur = u
  THEN obj_class = transfert_important  [Pos cover=15 Neg cover=0]
IF    agr_surface < 7200.00
  AND agr_rapport_surf_mais < 49.00
  AND 3.50 < agr_max_profondeur < 4.50
  AND agr_topo_forme_largeur = v
  THEN obj_class = transfert_important  [Pos cover=15 Neg cover=1]
IF    54800.00 < agr_surface < 124400.00
  AND agr_max_profondeur < 10.50
  AND agr_topo_forme_largeur = v
  THEN obj_class = transfert_important  [Pos cover=34 Neg cover=4]
IF    agr_pression < 453.50
  AND 15600.00 < agr_surface < 51600.00
  AND 3400.00 < agr_surf_max_mais < 7400.00
  AND agr_max_profondeur > 2.50
  THEN obj_class = transfert_important  [Pos cover=19 Neg cover=6]
IF    7000.00 < agr_surface < 21600.00
  AND agr_rapport_surf_mais < 42.00
  AND agr_rapport_disp_tampon < 2.50
  AND agr_max_profondeur > 2.50
  AND agr_topo_forme_denivele = plate
  THEN obj_class = transfert_important  [Pos cover=17 Neg cover=6]
IF    agr_rapport_prelevee < 90.50
  AND 1400.00 < agr_surface < 5000.00
  AND 39.00 < agr_rapport_surf_mais < 87.00
  THEN obj_class = transfert_important  [Pos cover=17 Neg cover=12]

[theorie transfert_faible]
IF    6200.00 < agr_surface < 15800.00
  AND 4.50 < agr_max_profondeur < 7.50
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=80]
IF    agr_rapport_risque_fort < 93.50
  AND agr_surface < 81000.00
  AND agr_rapport_surf_mais < 6.50
  AND agr_surf_max_mais < 1000.00
  AND agr_rapport_disp_tampon < 1.50
  THEN obj_class = transfert_faible  [Neg cover=1 Pos cover=74]
IF    agr_surface > 1800.00
  AND 10.00 < agr_rapport_surf_mais < 39.50
  AND agr_topo_forme_largeur = i
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=54]
IF    5000.00 < agr_surface < 28800.00
```

```
  AND agr_rapport_surf_mais > 26.50
  AND agr_surf_max_mais > 600.00
  AND agr_max_profondeur < 6.50
  AND agr_topo_forme_denivele = pentue
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=57]
IF    agr_pression > 143.00
  AND 42200.00 < agr_surface < 109400.00
  AND agr_rapport_surf_mais < 22.50
  AND agr_max_profondeur < 8.50
  AND agr_topo_forme_denivele = pentue
  AND agr_topo_forme_largeur = u
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=40]
IF    agr_surface < 84000.00
  AND agr_surf_max_mais < 5000.00
  AND agr_max_profondeur > 2.50
  AND agr_topo_forme_denivele = convexe
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=38]
IF    agr_pression > 28.00
  AND agr_rapport_prelevee < 93.50
  AND agr_surface > 4400.00
  AND 8.50 < agr_rapport_surf_mais < 77.00
  AND agr_surf_max_mais < 3400.00
  AND 2.50 < agr_max_profondeur < 3.50
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=31]
IF    agr_pression < 83.00
  AND 1800.00 < agr_surface < 6200.00
  AND agr_surf_max_mais < 2200.00
  AND agr_max_profondeur < 3.50
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=42]
IF    31200.00 < agr_surface < 53800.00
  AND agr_surf_max_mais < 3400.00
  AND agr_topo_forme_denivele = concave
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=36]
IF    agr_pression < 5481.50
  AND agr_surface > 27400.00
  AND 31.50 < agr_rapport_surf_mais < 35.50
  AND agr_surf_max_mais < 9600.00
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=20]
IF    agr_pression > 94.50
  AND agr_surf_max_mais < 4200.00
  AND agr_rapport_disp_tampon > 5.50
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=26]
IF    agr_pression < 181.50
  AND agr_rapport_risque_fort < 67.50
  AND 8.50 < agr_rapport_surf_mais < 38.50
  AND agr_surf_max_mais < 1600.00
  AND agr_topo_forme_denivele = pentue
  THEN obj_class = transfert_faible  [Neg cover=2 Pos cover=33]
IF    361.00 < agr_pression < 6263.00
  AND agr_rapport_risque_fort < 42.00
  AND agr_surface > 121800.00
  AND agr_surf_max_mais < 13400.00
  AND agr_max_profondeur < 15.00
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=20]
```

```
IF    agr_surface < 67800.00
  AND agr_rapport_surf_mais < 50.50
  AND 1800.00 < agr_surf_max_mais < 3400.00
  AND agr_topo_forme_denivele = plate
  THEN obj_class = transfert_faible  [Neg cover=0 Pos cover=21]
IF    agr_pression < 18517.50
  AND 73600.00 < agr_surface < 105200.00
  AND 24.50 < agr_rapport_surf_mais < 60.50
  AND agr_surf_max_mais < 29000.00
  THEN obj_class = transfert_faible  [Neg cover=4 Pos cover=16]
IF    agr_rapport_prelevee > 65.50
  AND 35.50 < agr_rapport_surf_mais < 67.50
  AND agr_max_profondeur < 3.50
  THEN obj_class = transfert_faible  [Neg cover=9 Pos cover=25]
IF    agr_pression < 2883.00
  AND 23000.00 < agr_surface < 67800.00
  AND 6.50 < agr_rapport_surf_mais < 51.50
  AND 2400.00 < agr_surf_max_mais < 9600.00
  AND agr_max_profondeur > 5.50
  THEN obj_class = transfert_faible  [Neg cover=4 Pos cover=33]
```

C.3 Règles résultant de la combinaison des deux apprentissages

Nous donnons ici l'ensemble des 34 motifs d'arbres d'exutoires résultant de l'apprentissage exposé dans la section 3.6.

```
[theorie transfert_important]
[Rule 1] [Pos cover = 95 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mais,D), eq(D,true),
 intro_noeud(A,C,[C,B],E), intro_noeud(A,C,[E,C,B],F), intro_noeud(A,C,[F,E,C,B],G),
 value(A,G,agr_topo_forme_denivele,H), eq(H,plate).
[Rule 2] [Pos cover = 78 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), value(A,B,itk_pression,C), lteq(C,2384.0),
 value(A,B,agr_rapport_disp_tampon,D), lteq(D,24.0).
[Rule 3] [Pos cover = 56 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,agr_topo_forme_largeur,D), eq(D,v),
 value(A,C,agr_surf_max_mais,E), gteq(E,12400.0).
[Rule 4] [Pos cover = 122 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), value(A,B,agr_surf_max_mais,C), gteq(C,15600.0),
 value(A,B,agr_rapport_risque_fort,D), gteq(D,2.0).
[Rule 5] [Pos cover = 74 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), value(A,B,agr_rapport_disp_tampon,C), lteq(C,6.0),
 value(A,B,agr_rapport_surf_mais,D), gteq(D,83.0), value(A,B,agr_max_profondeur,E),
 lteq(E,5.0).
[Rule 6] [Pos cover = 58 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), value(A,B,os_dispositif_tampon,C), eq(C,true),
```

```
 value(A,B,agr_surf_max_mais,D), gteq(D,10000.0), value(A,B,agr_rapport_disp_tampon,E),
 lteq(E,21.0).
[Rule 7] [Pos cover = 55 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mrt,D), gteq(D,51.25),
 intro_noeud(A,C,[C,B],E), intro_noeud(A,E,[E,C,B],F), intro_noeud(A,F,[F,E,C,B],G),
 intro_noeud(A,F,[G,F,E,C,B],H), intro_noeud(A,F,[H,G,F,E,C,B],I),
 value(A,I,agr_topo_forme_largeur,J), eq(J,i).
[Rule 8] [Pos cover = 42 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,agr_topo_forme_denivele,D),
 eq(D,plate), intro_noeud(A,C,[C,B],E), intro_noeud(A,C,[E,C,B],F),
 intro_noeud(A,B,[F,E,C,B],G), value(A,G,itk_type,H),
 eq(H,tout_en_post), value(A,G,os_pente,I), eq(I,nulle).
[Rule 9] [Pos cover = 37 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,agr_rapport_surf_mais,D), lteq(D,2.0),
 intro_noeud(A,C,[C,B],E), intro_noeud(A,B,[E,C,B],F), intro_noeud(A,B,[F,E,C,B],G),
 value(A,G,os_mais,H), eq(H,true), value(A,G,os_surface,I), gteq(I,0.12),
 value(A,G,os_mrt,J), lteq(J,5.625), value(A,G,agr_max_profondeur,K), lteq(K,2.0).
[Rule 10] [Pos cover = 69 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mais,D), eq(D,true),
 intro_noeud(A,B,[C,B],E), value(A,E,os_mrt,F), gteq(F,16.5), value(A,E,agr_surface,G),
 lteq(G,19200.0).
[Rule 11] [Pos cover = 16 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), value(A,B,agr_topo_forme_largeur,C), eq(C,v), value(A,B,os_pente,D),
 eq(D,nulle), value(A,B,os_mrt,E), lteq(E,9.75), value(A,B,agr_rapport_surf_mais,F),
 gteq(F,25.0), value(A,B,os_surface,G), gteq(G,0.2).
[Rule 12] [Pos cover = 62 Neg cover = 0]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D),
 intro_noeud(A,D,[D,C,B],E), intro_noeud(A,E,[E,D,C,B],F),
 intro_noeud(A,E,[F,E,D,C,B],G), intro_noeud(A,C,[G,F,E,D,C,B],H),
 value(A,H,agr_surf_max_mais,I), gteq(I,4400.0), value(A,H,agr_pression,J),
 lteq(J,1562.0), value(A,H,os_surface,K), lteq(K,0.68).
[Rule 13] [Pos cover = 16 Neg cover = 4]
transfert_important(A) :-
 racine(A,B), value(A,B,agr_surface,C), lteq(C,1600.0), value(A,B,os_mrt,D),
 gteq(D,-0.5), value(A,B,agr_rapport_risque_faible,E), gteq(E,31.0),
 value(A,B,agr_pression,F), gteq(F,14.0).
[Rule 14] [Pos cover = 62 Neg cover = 2]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_dispositif_tampon,D), eq(D,false),
 intro_noeud(A,C,[C,B],E), intro_noeud(A,E,[E,C,B],F), intro_noeud(A,F,[F,E,C,B],G),
 intro_noeud(A,G,[G,F,E,C,B],H), intro_noeud(A,G,[H,G,F,E,C,B],I),
 value(A,I,agr_surface,J), gteq(J,7600.0), value(A,I,agr_rapport_surf_mais,K),
 lteq(K,48.0), value(A,I,agr_max_profondeur,L), lteq(L,5.0), value(A,I,os_mrt,M),
 lteq(M,17.125).
[Rule 15] [Pos cover = 49 Neg cover = 1]
transfert_important(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), intro_noeud(A,C,[C,B],D), value(A,D,os_pente,E),
 eq(E,nulle), intro_noeud(A,C,[D,C,B],F), intro_noeud(A,C,[F,D,C,B],G),
```

```
 value(A,G,os_mrt,H), lteq(H,1.75), intro_noeud(A,B,[G,F,D,C,B],I),
 value(A,I,os_surface,J), gteq(J,0.4).
[Rule 16] [Pos cover = 22 Neg cover = 2]
transfert_important(A) :-
 racine(A,B), value(A,B,agr_topo_forme_largeur,C), eq(C,v),
 value(A,B,agr_rapport_disp_tampon,D), lteq(D,0.0), value(A,B,os_pente,E),
 eq(E,nulle), value(A,B,agr_max_profondeur,F), lteq(F,6.0),
 value(A,B,agr_rapport_surf_mais,G), gteq(G,24.0), value(A,B,os_mrt,H), gteq(H,22.75).

[theorie transfert_faible]
[Rule 1] [Pos cover = 23 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_topo_forme_denivele,C), eq(C,plate),
 value(A,B,agr_surface,D), lteq(D,3600.0), value(A,B,agr_surf_max_mais,E),
 lteq(E,800.0).
[Rule 2] [Pos cover = 17 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_rapport_disp_tampon,C), gteq(C,28.0).
[Rule 3] [Pos cover = 21 Neg cover = 1]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,agr_rapport_surf_mais,D),
 gteq(D,100.0), intro_noeud(A,B,[C,B],E), value(A,E,os_mrt,F), lteq(F,-0.5625).
[Rule 4] [Pos cover = 18 Neg cover = 1]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,agr_rapport_disp_tampon,D),
 gteq(D,6.0), value(A,C,agr_rapport_surf_mais,E), lteq(E,61.0).
[Rule 5] [Pos cover = 17 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), value(A,B,os_mrt,C), lteq(C,9.214286), value(A,B,os_surface,D),
 gteq(D,1.08), value(A,B,agr_rapport_prelevee,E), gteq(E,61.0).
[Rule 6] [Pos cover = 39 Neg cover = 2]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_topo_forme_denivele,C), eq(C,plate), value(A,B,os_mrt,D),
 lteq(D,6.125), value(A,B,agr_rapport_surf_mais,E), lteq(E,53.0).
[Rule 7] [Pos cover = 43 Neg cover = 6]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_topo_forme_largeur,C), eq(C,v),
 value(A,B,agr_rapport_surf_mais,D), lteq(D,18.0), value(A,B,os_surface,E),
 gteq(E,0.36).
[Rule 8] [Pos cover = 78 Neg cover = 9]
transfert_faible(A) :-
 racine(A,B), value(A,B,os_mrt,C), lteq(C,9.214286), value(A,B,agr_pression,D),
 lteq(D,922.0), value(A,B,os_surface,E), gteq(E,0.44).
[Rule 9] [Pos cover = 55 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,os_mrt,D), lteq(D,3.125),
 intro_noeud(A,C,[C,B],E), intro_noeud(A,E,[E,C,B],F), intro_noeud(A,F,[F,E,C,B],G),
 intro_noeud(A,G,[G,F,E,C,B],H), value(A,H,agr_topo_forme_denivele,I), eq(I,pentue).
[Rule 10] [Pos cover = 97 Neg cover = 1]
transfert_faible(A) :-
 racine(A,B), value(A,B,os_mrt,C), lteq(C,3.722222),
 value(A,B,agr_rapport_surf_mais,D), lteq(D,22.0), value(A,B,agr_surface,E),
 lteq(E,21600.0).
[Rule 11] [Pos cover = 23 Neg cover = 0]
```

```
transfert_faible(A) :-
 racine(A,B), value(A,B,os_mrt,C), lteq(C,-0.5), value(A,B,agr_rapport_surf_mais,D),
 lteq(D,69.0), value(A,B,agr_surface,E), gteq(E,3200.0).
[Rule 12] [Pos cover = 19 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_pression,C), lteq(C,113.0), value(A,B,os_pente,D),
 eq(D,nulle), value(A,B,os_mrt,E), gteq(E,20.25), value(A,B,agr_topo_forme_largeur,F),
 eq(F,u).
[Rule 13] [Pos cover = 35 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_surface,C), lteq(C,16000.0), value(A,B,os_mrt,D),
 lteq(D,9.0), value(A,B,os_surface,E), gteq(E,0.32), value(A,B,os_pente,F),
 eq(F,forte).
[Rule 14] [Pos cover = 18 Neg cover = 0]
transfert_faible(A) :-
 racine(A,B), intro_noeud(A,B,[B],C), value(A,C,agr_surface,D), lteq(D,19200.0),
 intro_noeud(A,C,[C,B],E), value(A,E,agr_surf_max_mais,F), gteq(F,6400.0),
 value(A,E,agr_topo_forme_denivele,G), eq(G,pentue).
[Rule 15] [Pos cover = 38 Neg cover = 1]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_topo_forme_largeur,C), eq(C,v),
 value(A,B,agr_surf_max_mais,D), lteq(D,1600.0), value(A,B,os_mrt,E), lteq(E,8.375),
 value(A,B,agr_rapport_surf_mais,F), gteq(F,20.0).
[Rule 16] [Pos cover = 29 Neg cover = 4]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_rapport_risque_faible,C), gteq(C,100.0),
 value(A,B,agr_pression,D), lteq(D,54.0), value(A,B,os_mrt,E), lteq(E,6.25),
 value(A,B,agr_topo_forme_largeur,F), eq(F,i).
[Rule 17] [Pos cover = 15 Neg cover = 5]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_rapport_risque_fort,C), lteq(C,0.0), value(A,B,os_mais,D),
 eq(D,false), value(A,B,os_mrt,E), gteq(E,57.599998), value(A,B,agr_surf_max_mais,F),
 lteq(F,13600.0).
[Rule 18] [Pos cover = 28 Neg cover = 7]
transfert_faible(A) :-
 racine(A,B), value(A,B,agr_surf_max_mais,C), lteq(C,4800.0), value(A,B,os_mais,D),
 eq(D,false), value(A,B,os_mrt,E), gteq(E,11.892858), value(A,B,agr_pression,F),
 gteq(F,562.0).
```

Bibliographie

Agrawal R., Imielinski T. & Swami A. N. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, p. 207–216 : ACM Press.

Alexander R. & Kelly T. (2006). Combining simulation with machine learning to build accident models. In *Proceedings of the 3rd International Workshop on Safety and Security in Multiagent Systems*, p. 1–5.

Ali K. M. & Pazzani M. J. (1993). HYDRA : A noise-tolerant relational concept learning algorithm. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, p. 1064–1071 : Morgan Kaufmann.

Allen J. F. (1981). An interval-based representation of temporal knowledge. In *Proceedings of the 7th International Conference on Artificial Intelligence (IJCAI'81)*, p. 221–226.

Bay S. D. & Pazzani M. J. (2001). Detecting group differences : Mining contrast sets. *Data Mining Knowledge Discovery*, **5**(3), 213–246.

Beven J. & Kirkby M. (1979). A physically based variable contributive area model of basin hydrology. *Hydrological Sciences Bulletin*, **24**, 43–69.

Blockeel H. & De Raedt L. (1998). Top-down induction of first-order logical decision trees. *Artificial Intelligence*, **101**(1-2), 285–297.

Breiman L., Friedman J., Ohlsen R. & Stone C. (1984). *Classification and regression trees.* Wadsworth International Group.

Brijs T., Vanhoof K. & Wets G. (2000). Reducing redundancy in characteristic rule discovery by using integer programming techniques. *Intelligent Data Analysis Journal*, **4**(3), 229–240.

Cerdan O., Souchere V., Lecomte V., Couturier A. & Le Bissonnais Y. (2001). Incorporating soil surface crusting processes in an expert-based runoff model : Sealing and transfer by runoff and erosion related to agricultural management. *CATENA*, **46**, 189–205.

Cestnik B. (1990). Estimating probabilities : A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI'90)*, p. 147–149.

Chen S. & Liu B. (2001). Generating classification rules according to user's existing knowledge. In *Proccedings of SIAM Conference on Data Mining (SIAM'01)*, p. 1–15.

Chi Y., Muntz R. R., Nijssen S. & Kok J. N. (2005). Frequent subtree mining - an overview. *Fundamenta Informaticae*, **66**(1-2), 161–198.

Clark P. & Boswell R. (1991). Rule induction with CN2 : Some recent improvements. In *Proceedings of the Fifth European Working Session on Learning (EWSL'91)*, p. 151–163 : Springer.

Clark P. & Niblett T. (1989). The cn2 induction algorithm. *Machine Learning*, **3**, 261–283.

Cordier M.-O. (2005). SACADEAU : A decision-aid system to improve stream-water quality. *ERCIM News. Special issue on Environmental Modelling*, (61), 35–36.

Cordier M.-O., Garcia F., C.Gascuel-Odoux, Masson V., Salmon-Monviola J., Tortrat F. & Trépos R. (2005a). A machine learning approach for evaluating the impact of land use and management practices on streamwater pollution by pesticides. In *Proceedings of International Congress on Modelling and Simulation (MODSIM'05)*, p. 2651–2657 : Modelling and Simulation Society of Australia and New Zealand.

Cordier M.-O., Masson V., Aurousseau P., Gascuel-Odoux C., Tortrat F., Falchier M., Heddadj D., Lebouille L., Garcia F. & Chanomordic B. (2005b). *Modélisation du transfert de pesticides dans un bassin versant en vue de la construction d'un outil d'aide à la décision pour la maîtrise de la qualité des eaux. Application au bassin versant du Frémeur (Morbihan).* Rapport interne, INRA - rapport interne "bv futur".

Cornuéjols A. & Miclet L. (2002). *Apprentissage artificiel : concepts et algorithmes.* Eyrolles.

De Carvalho F. (1994). Proximity coefficients between boolean symbolic objects. In *Proceedings of the 4th Conference of the International Federation of Classification Societies (IFCS'93)*, p. 387–394 : Springer-Verlag.

De Raedt L. & Dehaspe L. (1997). Clausal discovery. *Machine Learning*, **26**, 99–146.

De Raedt L. & Van Laer W. (1995). Inductive constraint logic. In *Proceedings of the 6th International Conference on Algorithmic Learning Theory (ALT'95)*, p. 80–94 : Springer-Verlag.

Dehaspe L. (1999). Frequent pattern discovery in first-order logic. *AI Communications*, **12**(1-2), 115–117.

Dehaspe L. & De Raedt L. (1996). DLAB : A declarative language bias formalism. In *Proceedings of the International Symposiumon Methodologies for Intelligent Systems*, volume 1079 of *LNAI*, p. 613–622 : SV.

Dehaspe L., Toivonen H. & King R. D. (1998). Finding frequent substructures in chemical compounds. In *4th International Conference on Knowledge Discovery and Data Mining*, p. 30–36 : AAAI Press.

Dolšak B. & Muggleton S. (1992). The application of inductive logic programming to finite-element mesh design. In *Inductive Logic Programming*, p. 453–472. Academic Press.

Duval B., Salleb A. & Vrain C. (2007). *On the Discovery of Exception Rules : A Survey*, In *Quality Measures in Data Mining*, p. 77–99. Springer in the Series Studies in Computational Intelligence.

Džeroski S. (1993). Handling imperfect data in inductive logic programming. In *Proceedings of the Fourth Scandinavian Conference on Artificial intelligence (SCAI'93)*, p. 111–125 : IOS Press.

Dzeroski S., Blockeel H., Kompare B., Kramer S., Pfahringer B. & Laer W. V. (1999). Experiments in predicting biodegradability. In *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP'99)*, p. 80–91 : Springer-Verlag.

Ferré S. & King R. D. (2005). A dichotomic search algorithm for mining and learning in domain-specific logics. *Fundamenta Informaticae – Special Issue on Advances in Mining Graphs, Trees and Sequences*, **66**, 1–32.

Frank E. & Witten I. H. (1998). Generating accurate rule sets without global optimization. In *Proceedings of 15th International Conference on Machine Learning (ICML'98)*, p. 144–151 : Morgan Kaufmann.

Friedman L. W. (1996). *The simulation metamodel.* Kluwer Academic Publishers.

Fromont E., Cordier M.-O., Quiniou R. & Hernandez A. (2003). Kardio and calicot : a comparison of two cardiac arrhythmia classifiers. In *Proceedings of AIME'03 Workshop : Qualitative and Model-based Reasoning in Biomedicine*, p. 29–33.

Fürnkranz J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, **13**(1), 3–54.

Fürnkranz J. & Flach P. (2003). An analysis of rule evaluation metrics. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, p. 202–209.

Gabriel T. R., Thiel K. & Berthold M. R. (2006). Rule visualization based on multi-dimensional scaling. In *IEEE International Conference on Fuzzy Systems.*

Gowda K. C. & Diday E. (1991). Symbolic clustering using a new dissimilarity measure. *Pattern Recognition*, **24**(6), 567–578.

Gupta G. K., Strehl A. & Ghosh J. (1999). Distance based clustering of association rules. In *Intelligent Engineering Systems Through Artificial Neural Networks (Proceedings of ANNIE 1999)*, volume 9, p. 759–764 : ASME Press.

Han J., Cai Y. & Cercone N. (1993). Data-driven discovery of quantitative rules in relational databases. *IEEE Transactions on Knowledge and Data Engineering*, **5**(1), 29–40.

Han J. & Cercone N. (2000). Ruleviz : a model for visualizing knowledge discovery process. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'00)*, p. 244–253 : ACM Press.

Horton R. (1938). The interpretation and application of runoff plot experiments with reference to soil erosion problems. In *Proceedings of Soil Science Society of America*, volume 3, p. 30–349.

Huber K.-P. & Berthold M. R. (1997). Simulation data analysis using fuzzy graphs. In *Proceedings of the Second International Symposium on Advances in Intelligent Data Analysis, Reasoning about Data (IDA '97)*, p. 347–358 : Springer-Verlag.

Ichino M. & Yaguchi H. (1994). Generalized minkowski metrics for mixed feature-type data analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, **24**(4), 698–708.

Inokuchi A., Washio T. & Motoda H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery (PKDD'00)*, p. 13–23.

Jaroszewicz S. & Simovici D. A. (2002). Pruning redundant association rules using maximum entropy principle. In *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'02)*, p. 135–147, London, UK : Springer-Verlag.

Kalbfleish J. (1979). *Probability and Statistical Inference*, volume 2. Springer-Verlag.

Kapur D. & Narendran P. (1986). Np-completeness of the set unification and matching problems. In *Proceedings of the 8th International Conference on Automated Deduction*, p. 489–495 : Springer-Verlag.

Karalič A. & Bratko I. (1997). First order regression. *Machine Learning*, **26**(2-3), 147–176.

King R. D., Whelan K. E., Jones F. M., Reiser P. G., Bryant C. H., Muggleton S. H., Kell D. B. & Oliver S. G. (2004). Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, **427**(6971), 247–252.

Kleijnen J. P. C. (1979). Regression metamodels for generalizing simulations results. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-9**(2), 93–96.

Laburthe F. (2000). Choco : implementing a cp kernel. In *Proceedings of Techniques foR Implementing Constraint programming Systems (TRICS'00)*, p. 118–133.

Lavrac N. & Dzeroski S. (1993). *Inductive Logic Programming : Techniques and Applications*. Routledge.

Lavrac N., Flach P., Kavsek B. & Todorovski L. (2002). Adapting classification rule induction to subgroup discovery. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, p. 266 : IEEE Computer Society.

Ma J. & Hayes P. (2006). Primitive intervals versus point-based intervals : Rivals or allies ? *Computer Journal*, **49**(1), 32–41.

MacQueen J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, p. 281–297 : University of California Press.

Malerba D., Esposito F., Gioviale V. & Tamma V. (2001). Comparing dissimilarity measures in symbolic data analysis. In *Joint Conferences on New Techniques and Technologies for Statistcs and Exchange of Technology and Know-how (ETK-NTTS'01)*, p. 473–481.

McSherry D. & Roantree D. (1999). Characteristic rule discovery in aurum-3. *Applied Intelligence*, **11**(3), 297–304.

Merkuryeva G. (2004). Metamodelling for simulation applications in production and logistics. In *Sim Serv Workshop : Roadmap of Simulation in Manufacturing and Logistics*, p. 1–6.

Michalski R.-S. (1973). Aqval/1 - computer implementation of a variable-valued logic system vl_1 and examples of its application to pattern recognition. In *Proceedings of the 1st International Joint Conference on Pattern Recognition (IJCPR'73)*, p. 3–17.

Michalski R. S., Mozetic I. & Hong J. (1986a). The multi-purpose incremental learning system aq15 and it testing application to three medical domains. In *Proceedings of AAAI-86*, p. 1041–1045.

Michalski S. R., Carbonell G. J. & Mitchell M. T. (1986b). *Machine learning an artificial intelligence approach*, volume 2. Morgan Kaufmann Publishers.

Mitchell T. (1980). *The need for biases in learning generalizations*. Rapport interne CBM-TR-117, Rutgers Computer Science Department.

Mitchell T. M. (1982). Generalization as search. *Artificial Intelligence*, **18**(2), 203–226.

Mladenic D., Bratko I., Paul R. J. & Grobelnik M. (1994). Using machine learning techniques to interpret results from discrete event simulation. In *Proceedings of the European Conference on Machine Learning (ECML'94)*, p. 399–402 : Springer-Verlag.

Mooney R. & Califf M. (1995). Induction of first-order decision lists : Results on learning the past tense of English verbs. In *Proceedings of the 5th International Workshop on Inductive Logic Programming*, p. 145–146 : Department of Computer Science, Katholieke Universiteit Leuven.

Muggleton S. (1995). Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, **13**(3-4), 245–286.

Muggleton S. & Buntine W. (1988). Machine invention of first order predicates by inverting resolution. In *Proceedings of 5th International Conference on Machine Learning (ICML '88)*, p. 339–351.

Muggleton S. & De Raedt L. (1994). Inductive logic programming : Theory and methods. *Journal of Logic Programming*, **19/20**, 629–679.

Muggleton S. & Feng C. (1990). Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory (ALT'90)*, p. 368–381 : Ohmsma, Tokyo, Japan.

Nakano S. & Uno T. (2003). *Efficient Generation of Rooted Trees*. Rapport interne nii-2003-005e, National Institute of Informatics, Tokyo.

Niblett T. (1987). Constructing decision trees in noisy domains. In *Proceedings of the second European Working Session on Learning (EWSL'87)*, p. 67–78.

Nienhuys-Cheng S.-H. & Wolf R. D. (1996). Least generalizations and greatest specializations of sets of clauses. *Journal of Artificial Intelligence Research*, **4**, 341–363.

Nilsson N. (1980). *Principles of Artificial Intelligence*. Tioga, Palo Alto.

Nédellec C., Rouveirol C., Adé H., Bergadano F. & B. T. (1996). *Advances in Inductive Logic Programming*, chapter Declarative Bias in Inductive Logic Programming, p. 82–103. IOS Press.

O'Keefe R. (1986). Simulation and expert systems- a taxonomy and some examples. *Simulation*, **46**(1), 10–16.

Parisay S. & Khoshnevis B. (1994). Automatic instance generation using simulation for inductive learning. In *Proceedings of the 26th conference on Winter simulation (WSC'94)*, p. 1409–1412 : Society for Computer Simulation International.

Pazzani M. J. & Kibler D. F. (1992). The utility of knowledge in inductive learning. *Machine Learning*, **9**, 57–94.

Pierreval H. (1992). Rule-based simulation metamodels. *European Journal of Operational Research*, **61**, 6–17.

Plotkin G. (1970). A note on inductive generalization. In *Machine Intelligence*, volume 5, p. 153–163. Edinburgh University Press.

Plotkin G. (1971). A further note on inductive generalization. In *Machine Intelligence*, volume 6, p. 101–124. Edinburgh University Press.

Quiniou R., Cordier M.-O., Carrault G. & Wang F. (2001). Application of ILP to cardiac arrhythmia characterization for chronicle recognition. *Lecture Notes in Computer Science*, **2157**, 220–227.

Quinlan J. (1986). Induction of decision trees. *Machine Learning*, **1**, 81–106.

Quinlan J. (1990). Learning logical definitions from relations. *Machine Learning*, **5**, 239–266.

Quinlan J. & Cameron-Jones R. (1993). Foil : A midterm report. In *Proceedings of the European Conference on Machine Learning (ECML'93)*, p. 3–20 : Springer-Verlag.

Quinlan J. R. (1987). Generating production rules from decision trees. In *Proceedings of the Tenth International Conference on Artificial Intelligence (IJCAI'87)*, p. 304–307 : Kaufmann.

Quinlan J. R. (1993). *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers.

Ras Z. W. & Tsay L.-S. (2003). Discovering extended action-rules, system dear. In *Intelligent Information Systems (IIS'03)*, p. 293–300 : Spinger-Verlag.

Ras Z. W. & Wieczorkowska A. (2000). Action-rules : How to increase profit of a company. In *European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'00)*, p. 587–592.

Rehm F., Klawonn F. & Kruse R. (2006). Rule classification visualization of high-dimensional data. In *Proceedings of Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'2006)*, p. 1944–1948.

Richards L. (1931). Capillary conduction of liquids in porous media. *Physics*, **1**, 318–333.

Sebag M. (1996). Delaying the choice of bias : A disjunctive version space approach. In *International Conference on Machine Learning (ICML'96)*, p. 444–452.

Shapiro E. Y. (1983). *Algorithmic Program Debugging*. MIT Press.

Silberschatz A. & Tuzhilin A. (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Trans. On Knowledge And Data Engineering*, **8**, 970–974.

Srinivasan A. (2003). Aleph manual, version 4 and above.

Srinivasan A. & Camacho R. (1999). Numerical reasoning with an ILP program capable of lazy evaluation and customised search. *Journal of Logic Programming*, **40**(2,3), 185–214.

Srinivasan A., Muggleton S., Sternberg M. J. E. & King R. D. (1996). Theories for mutagenicity : A study in first-order and feature-based induction. *Artificial Intelligence*, **85**(1-2), 277–299.

Torgo L. (1993). Controlled redundancy in incremental rule learning. In *Proceedings of European Conference on Machine Learning (ECML'93)*, volume 667 of *Lecture Notes in Computer Science*, p. 185–195 : Springer.

Tortrat F. (2005). *Modélisation orientée décision des processus de transfert par ruissellement et subsurface des herbicides dans les bassins versants agricoles.* PhD thesis, ENSA de Rennes, INRA-Agrocampus Rennes UMR Sol Agronomie Spatialisation Rennes-Quimper.

Tortrat F., Aurousseau P., Squividant H., Gascuel-Odoux C. & Cordier M.-O. (2004). Modèle numérique d'altitude (mna) et spatialisation des transferts de surface : utilisation de structures d'arbres reliant les exutoires de parcelles et leurs surfaces contributives. *Bulletin SFPT*, **n. 172**, 128–136.

Trépos R., Cordier M.-O., Masson V. & Gascuel C. (2007). Apprentissage de motifs spatiaux et agronomiques jouant un rôle dans la contamination de l'eau par les pesticides sur un bassin versant. In *8ème Rencontres des Jeunes Chercheurs en Intelligence Artificielle (RJCIA'07)*, p. 231–246 : Cépaduès.

Trépos R., Salleb A., Cordier M.-O., Masson V. & Gascuel C. (2005). A distance based approach for action recommendation. In *Proceedings of European Conference on Machine Learning (ECML'05)*, p. 425–434 : Springer Verlag.

Trépos R., Salleb A., Cordier M.-O., Masson V. & Gascuel C. (2006). Une approche fondée sur une distance pour la recommandation d'actions. In *Actes de 15ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle (RFIA'2006).* 9 pages.

Tsumoto S. & Hirano S. (2003). Visualization of rule's similarity using multidimensional scaling. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM '03)*, p. 339–346 : IEEE Computer Society.

Turmeaux T., Cassard D., Salleb A. & Vrain C. (2003). Apprentissage de règles caractéristiques. In *Extraction et Gestion des Connaissances (EGC'03)*, volume 17 of *Revue des Sciences et Technologies de l'Information - série RIA ECA*, p. 437–448 : Hermes Science Publications.

Valiente G. (2002). *Algorithms on Trees and Graphs.* Springer-Verlag.

Viaud V., Merot P. & Baudry J. (2004). Hydrochemical buffer assessment in agricultural landscapes from local to catchment scale. *Environmental Management*, **34**, 559–573.

Witten I. H. & Frank E. (2005). *Data Mining : Practical machine learning tools and techniques*. Morgan Kaufmann.

Wrobel S. (1997). An algorithm for multi-relational discovery of subgroups. In *Proceedings of First European Symposion on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, p. 78 – 87 : Springer Verlag.

Yang Q., Yin J., Ling C. X. & Chen T. (2003). Postprocessing decision trees to extract actionable knowledge. In *International Conference on Data Mining (ICDM'03)*, p. 685–688.

Zhang J., Bala J., Barry P. S., Meyer T. E. & Johnson S. K. (2002). Mining characteristic rules for understanding simulation data. In *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02)*, p. 381 : IEEE Computer Society.

Zhang J. & Michalski R. S. (1989). Rule optimization via the sg-trunc method. In *Proceedings of the Fourth European Working Session on Learning (EWSL'89)*, p. 251–262.

Zhao K., Liu B., Tirpak T. M. & Xiao W. (2005). Opportunity map : a visualization framework for fast identification of actionable knowledge. In *Proceedings of the 14th ACM international Conference on Information and Knowledge Management (CIKM'05)*, p. 60–67 : ACM.

Résumé

L'analyse des résultats de simulation d'un modèle représentant le fonctionnement d'un système environnemental est souvent difficile en raison du grand nombre de variables d'entrées et de la complexité des interactions entre processus modélisés.

Nous proposons d'analyser les résultats par des techniques d'apprentissage symbolique afin de produire des règles de classification utilisables pour l'aide à la décision. Deux approches pour l'apprentissage de règles sont proposées et comparées. Dans notre contexte, les objets à analyser sont des arbres dont les nœuds sont décrits par des attributs. La première approche génère des motifs d'arbres en utilisant la Programmation Logique Inductive. La seconde approche synthétise l'information contenue dans les arbres et induit des règles attribut-valeur.

Nous avons développé également un système d'aide à la décision qui suggère, à partir des règles induites, des actions permettant d'améliorer une situation proposée par l'utilisateur.

Ces contributions sont motivées par le projet SACADEAU qui a pour objectif de développer un système d'aide à la gestion des activités agricoles et des aménagements sur un bassin versant. Ce système s'appuie sur un modèle de simulation qui couple un modèle simulant le transfert de pesticides et un modèle simulant les pratiques agricoles liées à l'application de pesticides. La structure spatiale du bassin versant est représentée par un ensemble d'arbres d'exutoires alimentant le réseau hydrographique. Les techniques d'apprentissage proposées ont été expérimentées et comparées sur les données de cette application, puis intégrées dans un outil de visualisation.

Abstract

One often finds it difficult to analyze the results of a simulation model that represents the behavior of an environmental system. This is due to the large number of input variables and the complexity of interactions between the simulated processes.

We have proposed to use symbolic learning techniques in order to perform this analyze, the goal of which is to learn classification rules for decision support. Two rule-learning methods have been developed and compared. In our context, the objects to be analyzed are tree structures, the nodes of which are labelled by attributes. The first method, based on Inductive Logic Programming, generates tree patterns ; the second method synthesizes the information contained in the tree and induces attribute-value rules.

Afterwards, we have developed a system which, from induced rules, suggests actions so that a situation proposed by a user can be improved.

These contributions have been motivated by the SACADEAU project, devoted to develop a decision support system for the management of catchment areas. The project relies on a model that combines a model of farming practices with a model of pesticides transfer. The spatial structure of the catchment area is defined as a set of plot outlet trees feeding the stream. The proposed learning techniques have been tested and compared on this application, before having been incorporated into a visualization tool.