Caractérisation multiples minimales utilisant les formules booléennes partiellement définies

Fabien Chhel, Adrien Goëffon, Frédéric Lardeux, and Frédéric Saubion

LERIA, Université d'Angers, France
prenom.nom@univ-angers.fr

Résumé : Dans cet article, nous proposons de calculer un ensemble de formules booléennes partiellement définies qui correspond à la recherche de caractérisations de groupes de vecteurs booléens selon un ensemble de caractères (variables). Ce problème peut être utile pour de nombreuses applications pratiques où les données sont des résultats expérimentaux de tests de présence/absence. Ce problème a été largement étudié, mais généralement dans un cas plus limité, car ici nous avons à calculer simultanément un ensemble complet de formules de caractérisation minimale. Nous proposons donc deux approches différentes qui nous permettent d'étudier la satisfiabilité et la complexité sous-jacente de ce problème. Afin de mettre en évidence l'applicabilité de notre travail, nous fournissons aussi des résultats expérimentaux que nous avons obtenus à partir de données biologiques.

1 Introduction

L'apprentissage automatique de fonctions booléennes à partir d'exemples a été largement étudié pendant de nombreuses années, des travaux initiaux de Valiant (1984), Natarajan (1987), à d'autres plus récents comme Gavaldà & Thérien (2009). L'acquisition de connaissances comme des ensembles de vecteurs booléens est en effet un moyen facile de collecter les résultats d'expériences dans divers domaines d'application. Comme il sera illustré dans le présent document, les entités qui sont modélisées comme des vecteurs de booléens correspondent aux résultats d'une série de tests de diagnostic où la présence ou l'absence de caractères (les variables du problème) sont observées. L'objectif général est alors de fournir une formule booléenne qui est satisfaite par un ensemble d'entités (les exemples positifs) et qui est falsifiée par un autre ensemble (les exemples négatifs). La formule qui en résulte peut être exprimée en des syntaxes différentes (CNF, DNF, fonctions booléennes, BDD, ...). Les deux ensembles constituent une définition partielle de la formule. Bien entendu, de nombreuses autres techniques provenant de l'apprentissage peuvent être utilisées pour apprendre à partir d'exemples, telles que les classificateurs (par exemple, Support Vecteur Machine de Meyer *et al.* (2003)) mais elles n'apportent pas toujours des résultats précis et il est difficile de les utilisées sur un grand ensemble de variables. Le système FOIL de Quinlan (1990) a été conçu pour apprendre des clauses de Horn à partir d'exemples et peut

apparaître bien adapté à notre problème, mais la notion de minimalité n'est pas traitée avec ce système.

Dans cet article, nous nous concentrons sur une extension de ce problème initial en considérant simultanément plusieurs ensembles d'entités. Cette extension induit un supplément de niveau combinatoire. Informellement, on considère un ensemble $\mathcal{G}$ de n ensembles (appelés groupes) g_i d'entités. Chaque g_i doit être satisfait par une formule booléenne, et falsifié par toutes les autres entités qui sont dans $\mathcal{G} \setminus g_i$. Cela peut être interprété comme une caractérisation multiple exclusive des n groupes d'entités. En outre, nous exigeons que le nombre de variables utilisées dans chaque formule soit minimisé. La minimisation des expressions booléennes a sont importance, par exemple dans la conception de circuit électronique, et a sollicité beaucoup de travail afin de fournir des techniques d'optimisation efficaces. Néanmoins, notez que notre problème diffère du principal objectif de la conception de circuits électroniques dont le but est de réduire au maximum les composants logiques du circuit par rapport à un nombre fixe d'entrées et sorties (McCluskey (1956)). En effet, notre problème reviendrait à minimiser le nombre d'entrées en gardant le nombre de composants logiques fixe.

Nous étudions ici deux façons de traiter le problème qui correspondent à deux possibles formulations comme illustré par la figure 1. Le problème initial est défini par une matrice booléenne. Dans la partie gauche, nous abordons le problème par des techniques dont les formules booléennes sont le support. Nous étudierons d'abord la satisfiabilité du problème et sa complexité. Dans la partie droite, le problème se traduit par un problème de projection, basé sur l'approche de Makino *et al.* (1999), où les fonctions booléennes partiellement défini (*partially defined Boolean formula - pdBf*) sont exposées et dûment étudiées. Cela nous fournit une autre façon de calculer des solutions plus compactes.

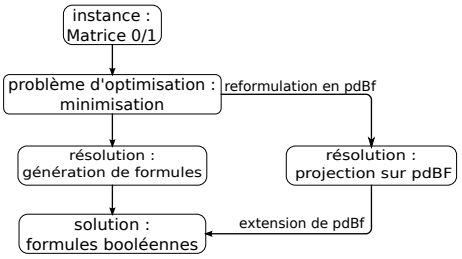

FIGURE 1 – Présentation générale des approches proposées.

La transformation entre les deux approches est polynomiale. Dans la première approche, les solutions sont exprimées sous forme de formules de la logique propositionnelle, qui induisent plusieurs difficultés liées à leur taille. En particulier, il parait difficile de contrôler les redondances et les tautologies lors de la recherche de solutions. En outre, l'espace de recherche peut être infini, sans définir des règles de réduction convergentes vers des formes normales. Au contraire, le second formalisme utilisant des projections fournit une représentation plus compacte des solutions. Néanmoins, notre première modélisation nous permet de bien étudier la satisfiabilité du problème et de

fournir une solution canonique.

Dans la dernière partie du document, nous montrons comment ce problème est appliquée à la biologie végétale avec des résultats expérimentaux. Ces expériences ont été menées avec des biologistes et ont conduit à l'élaboration d'un kit de diagnostic breveté pour l'identification des maladies bactériennes.

2 Le problème de caractérisation multiple

La description du problème de caractérisation multiple (PCM), dans le sens de la présence/absence de plusieurs caractères, nous permet naturellement d'utiliser le formalisme de la logique propositionnelle. Nous considérons qu'une matrice de booléens correspond à n caractères et m entités.

$$A \equiv \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

Chaque ligne de cette matrice représente une entité, caractérisée par la présence ou l'absence d'un ensemble de caractères. Nous considérons ensuite les caractères comme des variables booléennes et les entités comme des affectations booléennes.

Pour chaque colonne indexées par $j \in \{1, \dots, n\}$, nous définissons une variable propositionnelles x_i qui correspond à un caractère. X est l'ensemble initial des variables propositionnelles. Pour chaque ligne $i \in \{1, \dots, m\}$, nous considérons une entité e_i comme une interprétation booléenne, c'est à dire une application de X à $\{0, 1\}$ (faux, vrai), tel que $\forall i, j, e_i(x_j) = a_{ij}$. Nous notons $\mathcal{E}$ l'ensemble de toutes les entités. Pour une formule propositionnelle ϕ sur X et $e \in \mathcal{E}$, nous notons $e \models \phi$ le fait que l'interprétation e satisfasse la formule ϕ.

Remarquons que dans des cas réels, des techniques de pré-traitements sont applicables pour réduire la taille de la matrice originale. L'ensemble des variables qui est utilisé pour la caractérisation peut donc être plus petit que le jeu original. En particulier, s'il y a un caractère j tel que $\forall i, k \in \{1, \dots, m\}, a_{ij} = a_{kj}$, alors la colonne correspondante j est retirée de la matrice puisque ce caractère ne peut évidemment pas être utilisé pour distinguer les entités.

La définition des groupes correspond à des ensembles disjoints de lignes de la matrice A. S'il y a deux lignes identiques appartenant au même groupe, on peut supprimer l'une des eux. Dans ce cas, chaque groupe est alors un sous-ensemble des entités de $\mathcal{E}$. Notons que deux entités identiques peuvent appartenir à deux groupes différents. Nous étudierons ce dernier aspect en ce qui concerne la satisfiabilité du problème.

Un littéral est une variable $x \in X$ ou sa négation, notée $\neg x$. Une clause est une disjonction de littéraux. Nous notons $\mathcal{L}$ l'ensemble des littéraux construit sur X. Une formule ϕ est dite sous forme normale conjonctive (CNF) si elle est une conjonction de clauses. Une formule est dite en forme normale disjonctive (DNF) si elle est une disjonction de conjonctions de littéraux.

Nous allons maintenant définir une instance du problème PCM.

Définition 1 (Instance du PCM)
Une instance de PCM est définie par un tuple $(X, \mathcal{E}, G)$ où X est un ensemble de variables propositionnelles, $\mathcal{E}$ un ensemble d'entités défini sur X et $G \subseteq 2^{\mathcal{E}}$.

Nous nous focalisons maintenant sur la définition de la caractérisation d'un groupe qui doit permettre de reconnaître ses propres entités et à la discrimination (c.-à-d. de ne pas accepter) des entités d'autres groupes (d'un point de vue logique cela correspond donc à la satisfaction ou la réfutation de formules).

Définition 2 (Caractérisation de Groupe)
Soit une instance $(X, \mathcal{E}, G)$, une formule ϕ_g caractérise un groupe $g \in G$ ssi :
$\forall e \in g, e \models \phi_g$ *(accepte les entités du groupe)*
et
$\forall g' \in G \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g$ *(discrimine les entités des autres groupes).*

Par extension, nous notons $g \models \phi_g$ le fait que ϕ_g caractérise g selon la définition précédente. $Sol(g)$ est l'ensemble de toutes les caractérisations du groupe g et $Sol(g) = \{\phi_g | g \models \phi_g\}$.

Définition 3 (Solution du PCM)
Soit une instance $P \equiv (X, \mathcal{E}, G)$, une solution admissible du PCM est un $|G|$-tuple de formules $\Phi = (\phi_1, \cdots, \phi_{|G|})$ tel que $\forall i \in 1..|G|, g_i \in G$, $g_i \models \phi_i$.

$SOL(P)$ est l'ensemble de toutes les caractérisations multiples pour tous les groupes. $SOL(P) = Sol(g_1) \times \cdots \times Sol(g_{|G|})$. Soit un tuple de formules $\Phi = (\phi_1, \cdots, \phi_{|G|})$ et un ensemble de groupe G, nous notons par extension $G \models \Phi$ le fait que $\forall i \in 1..|G|, g_i \in G$, $g_i \models \phi_i$.

Définition 4 (Satisfiabilité du PCM)
Une instance $(X, \mathcal{E}, G)$ est satisfiable (resp. insatisfiable) ssi $\forall g \in G, Sol(g) \neq \emptyset$ (resp. $\exists g \in G, Sol(g) = \emptyset$).

Nous rappelons quelques définitions sur la taille et la longueur de formules.

Définition 5 (Taille et longueur d'une formule)
Soit ϕ une formule et $var(\phi)$ l'ensemble des variables de ϕ.
- *la taille de ϕ est $|var(\phi)|$;*
- *la longueur de ϕ, notée $len(\phi)$, est le nombre de feuilles du terme ϕ.*

Exemple 1
Soit $\phi = a \wedge b \wedge (\neg a \vee c)$, nous avons $|var(\phi)| = 3$ et $len(\phi) = 4$.

Définition 6 (Taille d'un tuple de formules)
Pour un tuple de formules $\Phi = (\phi_1, \cdots, \phi_n)$ nous avons $|\Phi| = |\bigcup_{\phi_i} var(\phi_i)|$.

3 L'approche par formules booléennes

Intuitivement, l'approche par formules booléennes semble la plus adaptée à résoudre le PCM.

3.1 Satisfiabilité

Notre premier but est d'étudier la satisfiabilité d'une instance du PCM (problème de décision). Nous proposons de construire une formule canonique qui peut être une solution pour n'importe quel problème satisfiable. Nous utilisons cette formule canonique pour exhiber une condition nécessaire et suffisante pour le problème de satisfiabilité.

Pour chaque entité e dans g, nous construisons la formule $\phi_e^+ \equiv \bigwedge_{x \in X} \delta(e,x)$ où $\delta : \mathcal{E} \times X \to \mathcal{L}$ est une fonction telle que $\delta(e,x) = \neg x$ si $e \models \neg x$ et $\delta(e,x) = x$ sinon. Nous notons $\phi_g^+ \equiv \bigvee_{e \in g} \phi_e^+$. Nous remarquons que ϕ_g^+ est en DNF. De la même façon nous avons $\phi_e^- \equiv \bigvee_{x \in X} \neg\delta(e,x)$ et $\phi_g^- \equiv \bigwedge_{g' \in G \setminus \{g\}} \bigwedge_{e' \in g'} \phi_{e'}^-$ (qui est en CNF). Nous notons $\phi_g \equiv \phi_g^+ \wedge \phi_g^-$.

Proposition 1 (Satifisfiabilité d'une instance)
Soit une instance $P \equiv (X, \mathcal{E}, G)$, nous avons :

$$SOL(P) \neq \emptyset \text{ ssi } \forall g \in G \begin{cases} \forall e \in g, e \models \phi_g \\ \text{et} \\ \forall g' \in G \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g \end{cases}$$

Preuve 1
Nous devons prouver que $L \Leftrightarrow R$ avec L et R respectivement la première partie et la seconde partie de la proposition.

$L \Leftarrow R$:
Si R nous avons immédiatement que l'ensemble $\{\phi_g\}_{g \in G}$ est une solution de P puisque chaque formule ϕ_g caractérise g (définition 2). Donc $SOL(P) \neq \emptyset$.

$L \Rightarrow R$:
Nous devons prouver que $\neg R \Rightarrow SOL(P) = \emptyset$. Considérons $\neg R$:

$$\begin{aligned} &\neg[\forall g \in G\big((\forall e \in g, e \models \phi_g) \wedge (\forall g' \in G \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g)\big)] \\ &\Leftrightarrow (\exists g \in G, \exists e \in g, e \not\models \phi_g) \vee (\exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \phi_g) \end{aligned}$$

Nous devons prouver que $(\exists g \in G, \exists e \in g, e \not\models \phi_g) \Rightarrow SOL(P) = \emptyset$ et $(\exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \phi_g) \Rightarrow SOL(P) = \emptyset$.

Commençons avec $\exists g \in G, \exists e \in g, e \not\models \phi_g \Rightarrow SOL(P) = \emptyset$. Nous avons $\phi_g \equiv \phi_g^+ \wedge \phi_g^-$. Par construction, $\forall g \in G, \forall e \in g, e \models \phi_g^+$. Alors $\exists g \in G, \exists e \in g, e \not\models \phi_g \Rightarrow SOL(P) = \emptyset$ ssi $\exists g \in G, \exists e \in g, e \not\models \phi_g^- \Rightarrow SOL(P) = \emptyset$.

$$\begin{aligned} &\exists g \in G, \exists e \in g, e \not\models \phi_g^- \\ &\Leftrightarrow \exists g \in G, \exists e \in g, e \not\models \bigwedge_{g' \in G \setminus \{g\}} \bigwedge_{e' \in g'} \phi_{e'}^- \\ &\Leftrightarrow \exists g \in G, \exists e \in g, \exists g' G \setminus \{g\}, \exists e' \in g', e \not\models \phi_{e'}^- \\ &\Leftrightarrow \exists g \in G, \exists e \in g, \exists g' G \setminus \{g\}, \exists e' \in g', e \not\models \bigvee_{x \in X} \neg\delta(e', x) \\ &\Leftrightarrow \exists g \in G, \exists e \in g, \exists g' G \setminus \{g\}, \exists e' \in g', e \models \bigwedge_{x \in X} \delta(e', x) \end{aligned}$$

Nous avons besoin du lemme suivant.

Lemme 1
$\forall e, e' \in \mathcal{E}, e \models \bigwedge_{x \in X} \delta(e', x) \Leftrightarrow e = e'$

Nous avons l'équivalence $\exists g \in G, \exists e \in g, \forall g' \in G \setminus \{g\}, \exists e' \in g', e = e'$ *ce qui implique* $SOL(P) = \emptyset$.

Considérons la seconde partie de $\neg R$. *Nous devons prouver que* $\exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \phi_g \Rightarrow SOL(P) = \emptyset$.

De la même manière par construction, $\forall g' \in G \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g^-$ *par* $\forall e' \in g', \phi_{e'}^- \equiv \bigvee_{x \in X} \neg\delta(e', x)$. *Considérons la partie correspondante à* ϕ_g^+ *dans* ϕ_g, *qui est :*

$$\begin{aligned}
&\exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \phi_g^+ \\
&\Leftrightarrow \exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \textstyle\bigvee_{e \in g} \phi_e^+ \\
&\Leftrightarrow \exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', \exists e \in g, e' \models \phi_e^+ \\
&\Leftrightarrow \exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', \exists e \in g, e' \models \textstyle\bigwedge_{x \in X} \delta(e, x) \\
&\Leftrightarrow \exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', \exists e \in g, e = e' \\
&\Rightarrow SOL(P) = \emptyset
\end{aligned}$$

Nous avons $\neg R \Rightarrow SOL(P) = \emptyset$.

Corollaire 1

Soit $P \equiv (X, \mathcal{E}, G)$ *:*

$$\exists g, g' \in G, g \neq g', \exists e \in g, \exists e' \in g', e = e' \Leftrightarrow SOL(P) = \emptyset$$

Remarquons que le corollaire précédent prouve que si G est une partition de $\mathcal{E}$, alors il existe toujours une solution. Bien sûr, il était manifeste qu'une instance possédant deux entités dans deux groupes différents n'était pas satisfiable mais les résultats précédents garantissent que c'est l'unique raison. Ce qui peut être reformulé par :

Corollaire 2

Soit $P \equiv (X, \mathcal{E}, G)$ *:*

$$P\ est\ satisfiable \Leftrightarrow \forall g, g' \in G, g \neq g', g \cap g' = \emptyset$$

Nous exhibons alors une solution canonique Φ_{Max}^P pour n'importe quel problème satisfiable P :

$$\Phi_{Max}^P \equiv (\phi_{g_1}, \ldots, \phi_{g_{|G|}})$$

Cette solution canonique est aussi une solution maximale.

Proposition 2

Soit $P \equiv (X, \mathcal{E}, G)$, *nous avons les propriétés suivantes :*

1. $SOL(P) \neq \emptyset \Leftrightarrow G \models \Phi_{Max}^P$
2. $\forall \Phi \in SOL(P), \Phi \leftrightarrow \Phi_{Max}^P$
3. $\forall \Phi \in SOL(P), |\Phi_{Max}^P| \geq |\Phi|$

Un algorithme par génération de formules booléennes a été développé mais il ne sera pas présenté dans ce papier. Nous exposons par contre les deux difficultés rencontrées par cette approche.

– La taille de la formule peut être infinie si nous nous basons sur les règles de la logique propositionnelle.

– La détection, lors de la construction,d'un terme redondant (commutativité des opérateurs logiques,distributions, ...) ou logiquement inutile (tautologie,insatisfiable) est très coûteuse.

Malheureusement ces problèmes ne sont pas totalement surmontés en se restreignant à une syntaxe de forme normale (CNF). Pour preuve, la génération des CNF sans redondances est semblable au problème de détection des clauses redondantes qui est coNP-complet (Boufkhad & Roussel (2000); Fourdrinoy *et al.* (2007)).

3.2 Caractérisation minimale

Avant d'étudier une autre formulation du PCM, nous introduisons un critère d'optimalité qui est de minimiser la taille de la solution. Étant donné un PCM P, nous cherchons d'abord à calculer une solution $(\phi_1, \cdots, \phi_n)$ telle que chaque ϕ_i est un élément minimal de $Sol(g_i)$. Remarquons qu'une solution minimale globale est un élément minimal de $SOL(P)$ (en ce qui concerne la définition 6) dont le calcul est beaucoup plus coûteux. En effet, une solution ne contenant que des formules minimales pour chacun des groupes ne garantit en rien l'obtention d'une solution minimale globale.

Nous nous intéressons maintenant à la caractérisation pour un groupe. Nous savons que les bornes sont données par :

– inférieure : $|min(SOL(P))| \geq max(|min(Sol(g_1))|, \cdots, |min(Sol(g_n)|)$.
– supérieure : $|min(SOL(P))| \leq |\Phi|$ avec $\Phi = (min(Sol(g_1)), \cdots, min(Sol(g_n))$

Sur la complexité, le problème de caractérisation minimale pour un groupe semblerait être NP-difficile, et le PCM minimale serait tout aussi difficile mais nous ne pouvons pas actuellement en apporter la preuve formelle. Mais en considérant ceux-ci comme un problème du choix d'une partie de X, l'idée est de se ramener au problème de couverture dont le problème de décision est bien connu pour être NP-complet (Garey & Johnson (1985) - le problème de minimisation est NP-difficile).

4 L'approche par *pdBf*

Dans cette section, nous proposons maintenant d'encoder le problème d'une manière différente afin d'exclure les problèmes majeurs à l'approche par formules booléennes (la possibilité de construire des termes infinis et la difficulté à éviter les redondances et tautologies) mais aussi de calculer directement une solution globale. Pour présenter l'approche par fonctions booléennes partiellement définies (*partially defined Boolean formula - pdBf*), nous rappelons quelques notations et concepts liés aux *pdBf* (voir Makino *et al.* (1999) pour plus de détails) et nous montrons l'équivalence entre les deux approches.

Définition 7 (Fonction booléenne)
Une fonction booléenne f est une application $f : \mathcal{B}^n \mapsto \mathcal{B}$ d'arité n dont $\mathcal{B}$ désigne l'ensemble des booléens.

Définition 8 (Fonction booléenne partiellement définie)
Une fonction booléenne partiellement définie pf est définie comme un sous-ensemble $\mathcal{C}^+ \cup \mathcal{C}^- \subseteq \mathcal{B}^n$, où $\forall e \in \mathcal{C}^+$ (resp. $\forall e' \in \mathcal{C}^-$), nous avons $pf(e) = \top$ (resp. $pf(e') =$

$\perp$) et $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) est appelé l'ensemble des exemples positifs (resp. négatifs). Une $pdBf_g$ est définie comme une pdBf *avec $\mathcal{C}^+ = g$ et $\mathcal{C}^- = G \setminus \{g\}$*

Nous étendons la notion de consistance de la logique propositionnelle aux *pdBf*.

Proposition 3 (Consistance et inconsistance d'une pdBf)
Une pdBf *est consistante (resp. inconsistante) ssi $\mathcal{C}^+ \cap \mathcal{C}^- = \emptyset$ (resp. $\mathcal{C}^+ \cap \mathcal{C}^- \neq \emptyset$).*

Définition 9 (Extension d'une pdBf)
Une extension d'une pdBf *est une formule ϕ, telle que ϕ est satisfaite (resp. falsifiée) par tous les éléments dans $\mathcal{C}^+$ (resp. $\mathcal{C}^-$)*

Dans la définition ci-dessus, on remarque que l'arité d'une *pdBf* est égale à la taille de son extension. Dans Eiter *et al.* (2002), les auteurs développent plusieurs techniques en temps polynomial pour trouver des extensions pour de nombreuses catégories définies. Nous introduisons maintenant la notion de projection qui est un concept clé pour calculer des représentations compactes de solutions.

Définition 10 (Projection)
Une projection est une fonction $\pi : \mathcal{B}^n \mapsto \mathcal{B}^k$ où $k \leq n$.

k est la dimension de la projection. π_{pdBf_g} est la projection associée à la $pdBf_g$. Nous notons $g \models \pi$ (i.e., π_{pdBf_g} est consistante) le fait que nous ayons :

$$\cup_{e \in \mathcal{C}^+} \pi_{pdBf_g}(e) \bigcap \cup_{e \in \mathcal{C}^-} \pi_{pdBf_g}(e) \neq \emptyset$$

Dans le but de préserver les résultats des sections précédentes, nous avons à prouver que les approches par formules et par projections (sur les *pdBf*) sont équivalentes.

Proposition 4 (Projetabilité et satisfiabilité)
$P \equiv (X, \mathcal{E}, G)$ est satisfiable ssi P est projetable sur G, c-à-d $\forall g \in G, \exists \phi, g \models \phi \Leftrightarrow \exists \pi, g \models \pi$

Preuve 2
$L \Rightarrow R$: En réutilisant la définition nous pouvons écrire $g \models \phi \Rightarrow \forall e \in g, e \models \phi_g \wedge \forall e' \notin g, e' \not\models \phi$ et en raisonnant par récurrence sur la longueur de ϕ, nous avons :

- *cas de base : $len(\phi) = 1$*
  Soit $\phi = x_i$ (resp. $\phi = \neg x_i$) et si $g \models \phi$ alors la seule pdBf *consistante x_i est $\forall e \in g, e(x_i) = \top$ (resp. $e(x_i) = \perp$) et $\forall e' \in G \setminus \{g\}, e'(x_i) = \perp$ (resp. $e(x_i) = \top$). Il existe alors π telle que $g \models \pi_{x_i}$.*
- *cas d'induction : Pour n>1, $len(\phi) = n - 1$ et $g \models \phi \Rightarrow g \models \pi$*
  Nous pouvons toujours construire $\phi' = \phi \otimes x_i$ avec $\otimes \in \{\wedge, \vee\}$, telle que $g \models \phi'$ (selon le principe du cas de base). Nous avons $\exists \pi', g \models \pi'$.

$R \Leftarrow L$: A partir d'une projection sur une $pdBf_g$, nous pouvons toujours trouver une solution via une extension (formule canonique par exemple).

Remarque 1

- *Une même projection qui est valide pour chaque groupe est donc une solution pour le PCM.*
- *Pour chaque groupe, l'extension trouvée peut-être de longueur non minimale mais nous garantissons par contre la minimalité de sa taille.*

L'algorithme EXACT-PROJ-CAR calcule des projections de la plus petite à la plus grande et retourne une extension pour la première projection trouvée (solution optimale).

5 Application : la minimisation de données biologiques

Le PCM est d'un grand intérêt pour de nombreux biologistes. La caractérisation précise des collections de souches bactériennes est un enjeu scientifique majeur, car les bactéries sont responsables d'importantes maladies sur les plantes, et donc soumise à des procédures officielles de contrôle (par exemple, en Europe, la directive 2000/29/CE). Le développement de tests de diagnostic est donc un enjeu important pour systématiquement identifier les souches de ces espèces. Dans ce contexte, le problème de la caractérisation correspond à l'identification d'un groupe de souches vis à vis d'autres groupes, basée sur la présence ou l'absence de certains caractères particuliers. Une souche est donc un vecteur de valeurs binaires qui reflète la présence (valeur 1) ou l'absence (valeur 0) de ces caractères. En outre, pour réduire le coût prohibitif des tests biologiques, nous nous intéressons à minimiser les solutions (le nombre de tests associés pour chaque identification) que ce soit pour un PCM ou pour un test plus spécifique pour un groupe donnée.

Les biologistes nous ont fourni les 3 instances suivantes :

- **A** : 8 groupes, 108 entités (de 2 à 54 par groupe), 155 caractères.
- **B** : 4 groupes, 112 entités (de 5 à 69 par groupe), 155 caractères.
- **C** : 7 groupes, 112 entités (de 2 à 40 par groupe), 155 caractères.

A, B et C sont des exemples provenant de l'API BioMérieux basée sur des propriétés bio-chimiques de l'espèce *Ralstonia*. Dans le tableau 1, nous présentons les résultats expérimentaux que nous avons obtenus avec notre algorithme. Les quatre premières lignes fournissent les caractéristiques de l'instance. La cinquième ligne donne la caractérisation minimale obtenue par EXACT-PROJ-CAR ("-" si inconnue) pour chacun des groupes. Nous n'avons aucune solution globale pour chacune de ces instances.

Instances	A								B				C						
Entités Totales	108								113				112						
Groupes	1	2	3	4	5	6	7	8	1	2	3	4	1	2	3	4	5	6	7
Entités par groupes	21	5	3	54	9	8	7	2	31	69	8	5	38	15	5	6	2	40	6
EXACT-PROJ-CAR (taille)	2	2	2	4	4	3	3	3	3	6	5	2	-	-	5	4	2	5	4

TABLE 1 – Application sur des instances réelles.

Les solutions de tailles inférieures à 3 sont instantanément trouvées. Nous constatons une augmentation exponentielle du temps de calcul qui peut aller de quelques secondes pour des tailles de 5, jusqu'à plusieurs heures pour des tailles supérieures.

Ces résultats, même partiels, ont permis d'apporter des solutions aux biologistes qui ne disposaient pas encore de telles méthodes pour analyser, traiter, et garantir la minimalité des caractérisations trouvées. De nouveaux tests de diagnostic ont ainsi pu être produits et vérifiés expérimentalement.

6 Conclusion et travaux futurs

Dans ce document, nous avons défini le problème de la caractérisation multiple et nous avons proposé deux formalismes afin d'étudier ses propriétés et de calculer une solution minimale. Malheureusement, l'approche complète associée aux *pdBf* pour un PCM échoue. Toutefois à partir du problème de caractérisation minimale des différents groupes, nous pouvons établir des bornes. Dans l'avenir, nous pensons nous orienter vers des méthodes de recherches locales ou évolutionnaires afin de trouver et améliorer les solutions de ce problème combinatoire et ainsi compléter l'approche par force brute de EXACT-PROJ-CAR. Une application aux données biologiques a été décrite pour mettre en évidence les utilisations possibles de ce problème d'acquisition de connaissances générales. Du point de vue des utilisateurs, le formalisme semble être utile, car il précise l'information (quels caractères sont importants et quelles formules sont à mettre en œuvre pour un test de diagnostic). Il serait également très utile d'être en mesure de calculer plusieurs solutions optimales différentes et les biologistes sont également intéressés par le découverte de relations entre les caractères. Les conséquences logiques seraient un moyen d'y parvenir. Nous pensons donc orienter nos prochains travaux vers des techniques issues de la compilation de connaissances.

Références

BOUFKHAD Y. & ROUSSEL O. (2000). Redundancy in random SAT formulas. In *Proc. of the 7th National Conference on AI and 20th Conference on Innovative Applications of AI*, p. 273–278 : AAAI Press / The MIT Press.

EITER T., IBARAKI T. & MAKINO K. (2002). Decision lists and related boolean functions. *Theor. Comput. Sci.*, **270**, 493–524.

FOURDRINOY O., GRÉGOIRE E., MAZURE B. & SAÏS L. (2007). Eliminating redundant clauses in SAT instances. In P. V. HENTENRYCK & L. A. WOLSEY, Eds., *The 4th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, p. 71–83 : Springer.

GAREY M. R. & JOHNSON D. S. (1985). *Computers and Intractability : a guide to the theory of NP-Completeness*. New York : Freeman.

GAVALDÀ R. & THÉRIEN D. (2009). An algebraic perspective on boolean function learning. In *Algorithmic Learning Theory, 20th International Conference, ALT 2009,Proceedings*, volume 5809 of *LNCS*, p. 201–215 : Springer.

MAKINO K., ICHI HATANAKA K. & IBARAKI T. (1999). Horn extensions of a partially defined boolean function. *SIAM J. Comput.*, **28**(6), 2168–2186.

MCCLUSKEY E. (1956). Minimization of boolean functions. *Bell System Tech. J*, **35**, 1417–1444.

MEYER D., LEISCH F. & HORNIK K. (2003). The support vector machine under test. *Neurocomputing*, **55**(1-2), 169–186.

NATARAJAN B. K. (1987). On learning boolean functions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, p. 296–304 : ACM.

QUINLAN J. R. (1990). Learning logical definitions from relations. *Machine Learning*, **5**, 239–266.

VALIANT L. G. (1984). A theory of the learnable. *Commun. ACM*, **27**(11), 1134–1142.