Application du problème de caractérisation multiple à la conception de tests de diagnostic pour la biologie végétale

Fabien Chhel, Frédéric Lardeux, Frédéric Saubion, Bruno Zanuttini

Application du problème de caractérisation multiple à la conception de tests de diagnostic pour la biologie végétale

Fabien Chhel Frédéric Lardeux Frédéric Saubion
Bruno Zanuttini *

Résumé

In this paper, we address a characterization problem coming from plant biology. The multiple characterization problem consists of entities (represented by Boolean assignments) belonging to several groups, and the goal is to find a characterization for each group using propositional logic. This characterization must be exact and the main difficulty is to compute the minimal one. We show that this problem is W[2]-Complete. We also propose a reformulation to a linear program and describe different approaches for its resolution. We experiment on random and real instances whose solutions are used to design a patent-protected diagnostic test.

Dans cet article, nous présentons un problème de caractérisation issu de la biologie végétale. Ce problème de caractérisation multiple consiste à traiter simultanément plusieurs groupes d'entités (représentées par des interprétations booléennes) et à trouver une caractérisation pour chacun d'eux sous forme de formules booléennes. Cette caractérisation doit être exacte et une des difficultés majeures est de trouver une caractérisation minimale. Une étude de complexité de ce problème est réalisée et nous arrivons à la conclusion que le problème est W[2]-Complet. Nous proposons une reformulation du problème en programmation linéaire ainsi que différentes approches de résolutions. Une étude expérimentale est réalisée sur des instances aléatoires ainsi que des instances réelles. Les résultats obtenus sur ces dernières ont pu être valorisés par le dépôt d'un brevet et la validation au niveau européen d'un test de dépistage basé sur les caractérisations trouvées.

*Laboratoire d'Étude et de Recherche en Informatique d'Angers, 2, boulevard Lavoisier F-49045 ANGERS Cedex 01, {chhel,lardeux,saubion}@info.univ-angers.fr/Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen, Boulevard du Maréchal Juin CS 14032, F-14032 CAEN cedex 05, bruno.zanuttini@unicaen.fr

1 Introduction

La plupart des bactéries appartenant au genre Xanthomonas sont responsables de pathologies sur une large gamme de cultures économiquement importantes, induisant notamment des pertes de rendement et diminuant ainsi la valeur marchande des semences. Quelques graines contaminées sont suffisantes pour générer une source d'inoculation primaire et occasionner ainsi une dissémination ultérieure plus large. En particulier, le pathovar [1] phaseoli de Xanthomonas Axonopodis (Xap) qui regroupe toutes les souches identifiées comme pathogènes sur le haricot [28], n'est pas endémique en Europe mais pour limiter son introduction, il est inscrit sur la liste des agents pathogènes de quarantaine.

La taxonomie du genre Xanthomonas n'est pas encore pleinement résolue, et la délimitation de certaines espèces dans ce genre fait encore débat [25] ; les approches phylogénétiques ne sont alors pas réellement applicables. Une approche possible pour l'identification des souches bactériennes consiste à utiliser un répertoire de gènes de virulence. Il s'agit ainsi de trouver la plus petite combinaison de gènes de virulence spécifiques au Xap. Cette combinaison peut être utilisée pour concevoir un test PCR multiplexe pour l'identification de Xap [2, 3], dont le coût est directement lié au nombre de gènes à tester. Les résultats obtenus montrent que la combinaison des tests moléculaires ainsi obtenus fournit une technique rapide pour l'identification de toutes les souches de Xanthomonas pathogènes sur les haricots. Nous abordons donc ce problème de la caractérisation de souches dans cet article.

Plus formellement, considérons un ensemble d'entités (les souches bactériennes) regroupées en groupes (les pathovars). Chaque entité est définie par la présence ou l'absence d'un ensemble de caractères (les gènes). Au regard de la représentation binaire qui est utilisée, une entité est considérée comme une interprétation booléenne sur les caractères, qui seront donc les variables booléennes du problème. Ainsi, pour chaque groupe, l'ensemble des entités fournit une table de vérité partielle d'une fonction booléenne vraie pour les interprétations correspondant aux entités du groupe et fausse pour toutes les autres entités des autres groupes. Une telle fonction sera appelée caractérisation d'un groupe. Ces fonctions booléennes seront représentées par des formules propositionnelles construites sur un langage fixé.

Le problème de caractérisation multiple consiste ainsi à trouver un ensemble de formules booléennes de sorte que chaque formule soit une caractérisation. Le terme multiple désigne le fait que nous considérons un ensemble de groupes dont les caractérisations sont dépendantes des entités contenues dans ceux-ci, mais aussi des entités appartenant aux autres groupes.

Dans la figure 1, nous avons 5 entités réparties dans 3 groupes et dont la description se base sur un ensemble de 3 caractères. Résoudre ce problème revient à caractériser chaque groupe. Il faut donc, pour chaque groupe, trouver une combinaison de variables permettant de construire une formule vraie pour toutes les entités du groupe et fausse pour les autres entités des autres groupes.

1. La notion de pathovar correspond à une subdivision du genre ayant des caractéristiques pathologiques observées communes.

Souches	Groupes	Caractères		
		a	b	c
e_1	g_1	0	0	0
e_2		0	0	1
e_3	g_2	1	1	1
e_4	g_3	1	1	0
e_5		0	1	0

FIGURE 1 – Exemple de problème de caractérisation multiple

Dans l'exemple de la figure 1, le groupe 1 est caractérisé par la négation des variables a et b alors que le groupe 2 est caractérisé par les variables b et c. Les souches du groupe 3 ont toutes en commun la négation de la variable c tout comme l'entité e_1 du groupe 1. Il faut donc ajouter une autre variable (b par exemple) pour être sûr de caractériser le groupe.

Dans cet article, après avoir défini formellement le problème de caractérisation multiple, nous étudions certaines de ses propriétés. Nous décrivons deux approches possibles de résolution, puis nous présentons une preuve sur l'inexistence d'un algorithme FPT (*fixed parameter tractable*) pour le problème de caractérisation multiple, sous les hypothèses communément admises. Nous étudions ensuite l'impact d'une transformation du problème vers le problème MIN-ONES afin de le traiter comme un programme linéaire. Enfin, nous proposons des résultats expérimentaux.

2 Le problème de caractérisation multiple

Nous présentons dans cette section le problème de caractérisation multiple ainsi que les travaux connexes.

2.1 Présentation du problème

Définition 1 (Instance du PCM) *Une instance du problème de caractérisation multiple est définie par un triplet $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ où $\mathcal{X}$ est l'ensemble des variables propositionnelles, $\mathcal{E}$ est l'ensemble des entités définies sur $\mathcal{X}$ et $\mathcal{G} \subseteq 2^{\mathcal{E}}$.*

Chaque entité représente une affectation booléenne, ou interprétation, définit ainsi $e : \mathcal{X} \to \{0, 1\}$, où 0 et 1 sont les valeurs de vérité respectivement fausse et vraie. $e(x)$ correspond donc à la valeur de vérité affectée à x dans l'interprétation e. Pour une formule propositionnelle ϕ quelconque sur $\mathcal{X}$, nous notons $e \models \phi$ le fait que l'interprétation $e \in \mathcal{E}$ satisfait la formule ϕ. Une entité $e \in \mathcal{E}$ sera classiquement représentée par un n-uplet de valeur booléennes (c'est-à-dire un élément de $\{0, 1\}^n$). Ainsi, une instance $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ peut être vue comme une matrice booléenne (a_{ij}) dont les n colonnes correspondent aux variables boo-

léennes de $\mathcal{X}$ et les m lignes aux entités de $\mathcal{E}$. Chaque variable x_j correspond à la colonne $j \in \{1, \ldots, n\}$. Chaque entité e_i correspond à une ligne $i \in \{1, \ldots, m\}$.

Nous pouvons appliquer des prétraitements pour réduire la taille de la matrice en réduisant le nombre de variables réellement utiles et/ou le nombre d'entités. Nous définissons ainsi la notion d'instance non redondante.

Définition 2 (Instance non redondante) *Une instance est non redondante ssi :*

- *Il n'existe pas de colonnes dont toutes les valeurs sont identiques :* $\nexists j \in \{1, \ldots, n\}$ *tel que* $\forall i \in \{1, \ldots, m\}, a_{ij} = 1$ *ou* $\forall i \in \{1, \ldots, m\}, a_{ij} = 0$ *;*
- *Chaque colonne est unique :* $\nexists j, k \in \{1, \ldots, n\}, j \neq k$ *tels que* $\forall i \in \{1, \ldots, m\}, a_{ij} = a_{ik}$ *;*
- *Chaque entité est unique :* $\nexists i, \ell \in \{1, \ldots, m\}, i \neq \ell$ *tels que* $\forall j \in \{1, \ldots, n\}, a_{ij} = a_{\ell j}$.

L'application d'un algorithme d'élimination de la redondance s'effectue au pire des cas en temps $\mathcal{O}(|\mathcal{X}||\mathcal{E}|)$ en triant la matrice selon les lignes et les colonnes, et peut conduire à une instance mal-formée, c'est-à-dire dont toutes les colonnes ont été supprimées ou possédant des groupes vides.

Une fois l'instance réduite, nous pouvons préciser la notion de solution d'un problème de caractérisation multiple.

Définition 3 (Caractérisation d'un groupe) *Pour une instance* $(\mathcal{X}, \mathcal{E}, \mathcal{G})$*, une formule* ϕ_g *caractérise un groupe* $g \in \mathcal{G}$ *ssi :* $\forall e \in g, e \models \phi_g$ *(acceptation des entités du groupe) et* $\forall g' \in \mathcal{G} \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g$ *(rejet des entités des autres groupes).*

Par extension, nous notons $g \models \phi_g$ le fait que ϕ_g caractérise g selon la définition précédente. $Sol(g)$ représente l'ensemble des solutions d'un groupe g. $Sol(g) = \{\phi_g | g \models \phi_g\}$.

Définition 4 (Solution d'un PCM) *Pour une instance* $(\mathcal{X}, \mathcal{E}, \mathcal{G})$*, une solution admissible du PCM est un* $|\mathcal{G}|$*-uplet de formules* $\Phi = (\phi_1, \cdots, \phi_{\mathcal{G}})$ *tel que* $\forall i \in \{1, \ldots, |G|\}, g_i \in \mathcal{G}$*,* $g_i \models \phi_i$.

Soit $\mathcal{I} = (\mathcal{X}, \mathcal{E}, \mathcal{G})$, $SOL(\mathcal{I})$ est l'ensemble de toutes les solutions multiples pour tous les groupes. $SOL(\mathcal{I}) = Sol(g_1) \times \cdots \times Sol(g_{|\mathcal{G}|})$, où $\times$ désigne le produit cartésien. Pour un ensemble de groupes $\mathcal{G}$ et un $|G|$-uplet de formules $\Phi = (\phi_1, \cdots, \phi_{|\mathcal{G}|})$, nous notons par extension $\mathcal{G} \models \Phi$ le fait que $\forall i \in \{1, \ldots, |G|\}, g_i \in \mathcal{G}$, $g_i \models \phi_i$.

Définition 5 (Satisfiabilité d'un PCM) *Une instance* $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ *est satisfiable (resp. insatisfiable) ssi* $\forall g \in \mathcal{G}, Sol(g) \neq \emptyset$ *(resp.* $\exists g \in \mathcal{G}, Sol(g) = \emptyset$*).*

Il est évident que toute instance possédant deux interprétations identiques dans deux groupes différents n'est pas satisfiable d'où :

Proposition 6 *Une instance* $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ *est satisfiable ssi* $\forall g, g' \in \mathcal{G}, g \neq g', g \cap g' = \emptyset$.

Une conséquence immédiate est qu'une instance non-redondante et bien formée est satisfiable. En pratique, le test de satisfiabilité est effectué implicitement lors de l'étape de prétraitement et à charge de l'utilisateur de modifier ou supprimer les entités mises en cause dans l'échec de ce test.

Définition 7 (Taille d'un n-uplet de formule) *Pour un n-uplet* $\Phi = (\phi_1, \cdots, \phi_n)$ *nous avons* $|\Phi| = |\bigcup_{\phi_i} var(\phi_i)|$, *où* $var(\phi)$ *retourne l'ensemble des variables apparaissant dans* ϕ.

Définition 8 (k-PCM (problème de décision)) *Soit une instance* $\mathcal{I}$. *Une caractérisation multiple minimale* k *est un ensemble de formules* $\Phi \in Sol(\mathcal{I})$ *vérifiant* $|\Phi| \leq k$ *avec* $k \in \mathbb{N}^+$.

Le problème de caractérisation multiple minimale pour une taille k ne correspond pas nécessairement à une solution minimale de $(\phi_1, \cdots, \phi_n)$ telle que chaque ϕ_i est un élément minimal de $Sol(g_i)$. Nous définissons le problème d'optimisation comme suit.

Définition 9 (MIN-PCM (problème d'optimisation)) *Pour une instance* $\mathcal{I}$, *une caractérisation optimale multiple minimale est un ensemble de formules* $\Phi^* \in Sol(\mathcal{I})$ *vérifiant* $|\Phi^*| \leq |\Phi|$ *pour tout* $\Phi \in Sol(\mathcal{I})$.

2.2 Travaux existants

Ce problème de caractérisation multiple pourrait sembler facile à résoudre au moyen d'approches existantes. Plusieurs techniques, basées sur les travaux initiaux de MacCluskey [18], ont été développées pour l'élaboration de circuits électroniques dont nous cherchons à minimiser le nombre de portes logiques. Il ne s'agit pas de minimiser le nombre de variables qui correspondent aux entrées de ces composants mais le nombre de connecteurs logiques de la fonction booléenne de caractérisation. Ces méthodes se basent sur la notion d'impliquants premiers car les fonctions booléennes des circuits logiques programmables sont codées généralement sous forme normale disjonctive (DNF). Dans [26], l'auteur montre que le problème de la minimisation du nombre des impliquants premiers est Σ_p^2-complet. Nous pouvons aussi faire un rapprochement avec les méthodes de validation des composants électroniques connues sous l'acronyme anglais *ATPG - Automatic test pattern generation* - qui consiste à spécifier un ensemble de tests pour la validation des états de sortie en fonction des états d'entrée et des portes logiques du circuit, mais nous devons prendre une contrainte temporelle, ce qui diffère du PCM.

D'autres approches basées sur les techniques de l'apprentissage automatique pourraient également être envisagées.

– Techniques d'apprentissages supervisées : Les séparateurs à vaste marge (SVM) [19], réseaux bayésiens ou de neurones et arbres de décisions sont généralement utilisés comme classifieurs et impliquent un paramétrage sur un jeu de données test en vue de classifier tout nouvel échantillon. Pour le PCM, la prédiction se pose de manière différente car nous ne nous restreignons qu'aux entités de nos groupes, considérées comme la connaissance exhaustive du problème. Ainsi, nous ne nous intéressons pas à la généralisation du classifieur et nous cherchons seulement à identifier les entités décrites dans nos données. De plus une instance du PCM peut contenir des classes non linéairement séparables en raison de la présence de fonctions booléennes équivalentes à l'opérateur XOR (ou-exclusif), ce qui écarte toute méthode utilisant des régressions linéaires.
– Apprentissage automatique de fonctions booléennes : L'apprentissage de fonctions booléennes à partir d'exemples a été intensivement étudié dans [15, 21, 11]. Je supprimerais la suite... Un résultat important est qu'il est possible d'apprendre une expression booléenne en un temps polynomial lorsque notre modèle d'apprentissage est probablement approximativement correct (PAC) selon [27]. Une version de FOIL [24] pour la logique propositionnelle permet de construire inductivement une formule de Horn en essayant de valider les exemples positifs et à réfuter les exemple négatifs. Toutefois, le PCM défini ici ne répond pas à ce critère.... et la remplacerais par ... Ce cadre correspond directement à l'identification de fonctions booléennes consistantes avec des exemples positifs et négatifs. Néanmoins, à notre connaissance aucun travail de ce domaine n'est directement applicable au problème PCM, qui requiert d'une part de considérer plusieurs groupes simultanément (et non deux), et d'autre part de chercher une (petite) caractérisation parmi plusieurs possibles, plutôt qu'une fonction cible fixée. Toutefois, les travaux les plus proches sont certainement ceux sur l'apprentissage de k-juntas [20]... fin proposition modif.
– Dans le domaine de la fouille de données, nous pouvons situer aussi le PCM par rapport aux travaux relatifs aux *feature extraction* [13] qui permettent de réduire les dimensions des données traitées afin de ne garder que l'information pertinente. Dans ce contexte, le PCM serait défini par un problème non-linéaire, sans marge d'erreur et multivarié puisqu'une solution est définie sur un ensemble de variables ayant un domaine discret (ici binaire). Les techniques basées sur l'extraction de motifs émergents [1] et plus particulièrement pour les données binaires [23] s'appliquent aux problèmes non-supervisés pour lesquels aucune hypothèse sur la structuration des données n'est connue à l'avance, ce qui ne correspond pas au PCM dont chaque entité appartient à un groupe.

Dans l'ensemble, ces méthodes ne garantissent pas la minimalité de l'expression apprise, ou tolèrent une marge d'erreur. Ainsi, elles ne répondent que partiellement à la résolution du PCM.

3 Résolution

Nous proposons plusieurs approches de résolution basées sur les fonctions booléennes partiellement définies. L'objectif n'est pas de proposer l'algorithme le plus efficace pour le PCM mais plutôt de présenter différentes méthodes de résolution afin de pouvoir par la suite observer leurs comportements pour des instances aux propriétés variées (section 6.2).

3.1 Résolution basée sur les fonctions booléennes partiellement définies (pdBfs)

Informellement, nous cherchons à minimiser le nombre de colonnes k en essayant de garder satisfiable l'instance réduite du PCM. Nous utilisons les fonctions booléennes partiellement définies, *partially defined Boolean formula - pdBfs*, [17] qui permettent de proposer un cadre de résolution intéressant.

Une pdBf est vue comme une fonction booléenne pour laquelle certaines interprétations ne sont pas définies. La classe C^+ (resp. C^-) désigne l'ensemble des exemples positifs (resp. négatifs). À partir de toute fonction booléenne, nous pouvons calculer une formule caractérisant l'ensemble des interprétations, appelée extension. Il en est de même pour les pdBfs où une DNF (formule en forme normale disjonctive) est une extension facilement calculable caractérisant la classe $\mathcal{C}^+$.

En ce qui concerne le PCM, nous construisons un ensemble de pdBfs emboîtées où chaque classe C^- est l'union des classes C^+ des autres groupes. Nous nous appuyons sur la notion de projection pour calculer de nouvelles solutions.

Définition 10 (Projection) *Une projection π d'une instance $\mathcal{I}$ de PCM est la donnée d'un sous-ensemble $\mathcal{X}' \subseteq \mathcal{X}$, définissant implicitement l'instance $\mathcal{I}' = (\mathcal{X}', \{\pi(e) \mid e \in \mathcal{E}\}, \{\pi(g) \mid g \in \mathcal{G}\})$, où $\pi(e)$ est la restriction de e aux variables de $\mathcal{X}'$, et $\pi(g)$ est $\{\pi(e) \mid e \in g\}$. On appelle* dimension *d'une projection π le cardinal de $\mathcal{X}'$.*

La minimisation du nombre de colonnes pour le problème PCM revient à chercher la projection satisfiable de plus petite dimension associée aux pdBfs de chaque groupe.

La principale difficulté du PCM ne dépend pas de la structure des formules de l'ensemble solution mais du choix des variables présentes dans celui-ci. La satisfiabilité d'un ensemble de formules de taille k pour une instance du PCM est équivalente au fait que nous cherchons un sous-ensemble de pdBfs consistantes, issues d'une projection satisfiable de dimension k sur les pdBfs initiales.

3.2 Approche de résolution complète

Le solveur *Exact-Proj-Car*, proposé dans [4], est basé sur une approche complète consistant à valider ou non la présence d'une caractérisation de taille k. Deux types d'exploration sont possibles :

– Une exploration en largeur consistant à montrer qu'il n'existe aucune caractérisation valide de taille k avant de tester celles de taille $k+1$. En commençant avec $k = \lceil log_2(|\mathcal{G}|) \rceil$ nous garantissons donc de trouver la caractérisation valide optimale. En effet, il est impossible de distinguer plus de 2^k classes avec une projection de dimension k.
– Une exploration en profondeur partant de la caractérisation maximale (toutes les variables) et recherchant une caractérisation valide de taille $k-1$ dès qu'une caractérisation valide de taille k est trouvée.

Ces deux approches garantissent toutes les deux de trouver une caractérisation valide optimale mais en cas d'arrêt de la recherche (temps limite atteint...), seule l'exploration en profondeur est capable de fournir une caractérisation valide. De plus, en terme de nombre de caractérisations testées, l'exploration en largeur semble la plus coûteuse (hormis pour les solutions de très petite taille). En effet, il est évident que l'exploration en largeur traitera au moins $\binom{n}{k}$ caractérisations (n étant le nombre de variables totales) alors que pour l'exploration en profondeur il est impossible de prévoir ce nombre (au mieux $n-k$ et au pire $\sum_{i=n}^{i=k} \binom{n}{i}$). Remarquons que l'utilisation conjointe de ces deux méthodes permet de borner la caractérisation optimale.

Ces explorations ont toutes les deux des choix de variables à faire. Pour cela nous avons proposé une heuristique de branchement basée sur un classement des variables de manière statique au début de la recherche. L'ordre utilisé est un calcul d'entropie inspiré par la technique proposée dans [7] qui privilégie les variables permettant de séparer un groupe par rapport aux autres.

3.3 Approche de résolution par recherche locale

Comme nous le verrons dans la section suivante, utiliser une approche complète peut s'avérer inefficace dans le cas de PCM de grande taille. Une alternative est donc de résoudre le problème à l'aide d'un algorithme de recherche locale [16]. L'optimalité des résultats n'est plus garantie mais une solution de bonne qualité est généralement trouvée assez rapidement.

Le principe de notre approche, appelée *LS-Proj-Car*, est de parcourir l'espace des projections valides en utilisant une liste tabou [12] afin de limiter les risques de cycles durant la recherche. La fonction de voisinage est définie par l'ensemble des projections ayant une variable de plus ou de moins que la projection actuelle. Le passage d'un voisin à un autre se fait à l'aide de deux opérateurs : *ajout_var* qui ajoute aléatoirement une variable à la projection et *supprime_var* qui supprime la première variable permettant d'atteindre une projection valide. La recherche commence avec la projection de dimension maximale (toutes les variables) et applique *supprime_var* à chaque fois que cela est possible. La liste tabou permet d'éviter d'ajouter et de supprimer une variable plusieurs fois de suite.

Afin de permettre une exploration plus large de l'espace de recherche, plusieurs redémarrages de la recherche sont effectués. Pour ne pas reparcourir les mêmes projections, un mécanisme d'apprentissage permet de garantir un début de recherche différent pour chaque relance. Son principe est de donner un poids

inversement proportionnel à l'ordre de sélection lors des recherches précédentes, pour chacune des variables, afin de pouvoir initier les relances suivantes en sélectionnant des variables ayant un poids faible. Ainsi, chaque relance oriente la recherche vers des zones de l'espace de recherche non explorées.

4 Complexité et monotonie

L'étude de la complexité du k-PCM montre les limites algorithmiques pour la résolution de ce problème.

4.1 Complexité

Dans cette section, nous caractérisons la complexité du problème. Nous rappelons tout d'abord quelques définitions sur la complexité paramétrée [10].

Définition 11 (Langage paramétré) *Un langage paramétré est un langage $\mathcal{L} \subseteq \Sigma^* \times \mathbb{N}$, où Σ désigne un alphabet fini et $\mathbb{N}$ l'ensemble des entiers naturels. Pour $\langle x, k\rangle \in \Sigma^* \times \mathbb{N}$, k est appelé le paramètre associé aux données x.*

Nous considérons maintenant les problèmes de décision paramétrés, définis comme suit :

Définition 12 (Problème traitable avec paramètre fixé (fixed-parameter tractable - FPT)) *Un problème P est traitable avec paramètre fixé si l'appartenance de $\langle x, k\rangle \in \mathcal{L}_P$ est décidable en temps $\mathcal{O}(f(k)|x|^c)$, avec c une constante et f une fonction $\mathbb{N} \mapsto \mathbb{N}$.*

L'idée importante est que si un problème est FPT, il est possible de « confiner » la partie exponentielle dépendant du paramètre k, et ainsi la taille de l'entrée n'influence que polynomialement le temps de résolution. Le problème FPT le plus connu est sans doute le problème de couverture VERTEX-COVER [9]. Comme pour la théorie de la NP-complétude, une réduction spécifique est définie pour la complexité paramétrée, appelée FPT-réduction, qui permet de démontrer qu'un problème est aussi difficile qu'un autre. Intuitivement, une FPT-réduction est similaire à une réduction standard (many-one, polynomiale), mais impose que le paramètre de l'image de la réduction soit indépendant de la taille de l'instance initiale (seulement de son paramètre).

Définition 13 (FPT-réduction) *$\mathcal{L}$ se réduit à $\mathcal{L}'$ par une réduction $\preccurlyeq_{fpt}$, ce qu'on note $\mathcal{L} \preccurlyeq_{fpt} \mathcal{L}'$, s'il existe deux fonctions $f, f' : \mathbb{N} \mapsto \mathbb{N}$ et une fonction $\lambda : \langle x, k\rangle \mapsto x'$ définie sur $\Sigma^* \times \mathbb{N} \mapsto \Sigma^*$, telles que λ soit calculable en $\mathcal{O}(f'(k)|x|^c)$ et $\langle x, k\rangle \in \mathcal{L}$ ssi $\langle \lambda(x), f(k)\rangle \in \mathcal{L}'$.*

Il est possible de définir un ensemble de classes de complexité paramétrée closes par la FPT-réduction, définies à l'aide d'une WEFT-hiérarchie [8] de la forme suivante : $FPT \subseteq W[1] \subseteq W[2] \subseteq ... \subseteq XP$ dont tout problème d'un certain degré est réductible vers un problème canonique de satisfaction sur des

circuits de décisions pondérés. Nous ne nous intéressons pas à la définition de cette hiérarchie mais nous utilisons l'hypothèse admise que cette échelle est propre et ne s'effondre pas. Ceci implique que les inclusions sont strictes entre les différentes classes.

Nous allons utiliser la FPT-réduction et le fait que le problème SET-COVER soit de la classe de complexité paramétrée W[2] pour obtenir la complexité du PCM.

Proposition 14 *PCM et SET-COVER sont équivalents par FPT-réduction.*

Proof *Rappelons tout d'abord la définition du problème SET-COVER paramétré : étant donnée une collection $\mathcal{C}$ de sous-ensembles d'un ensemble $\mathcal{U}$ et un paramètre $k \in \mathbb{N}^+$, on cherche à savoir s'il existe une sous-collection $\mathcal{C}' \subseteq \mathcal{C}$ de taille $|\mathcal{C}'| \leq k$ et vérifiant $\bigcup_{c \in \mathcal{C}'} = \mathcal{U}$.*

Montrons tout d'abord PCM $\preccurlyeq_{fpt}$ SET-COVER.

Soit $\langle \mathcal{I}, k \rangle = \langle (\mathcal{X}, \mathcal{E}, \mathcal{G}), k \rangle$ une instance du problème PCM paramétré. Chaque paire d'entités $\{e, e'\} \in \mathcal{E}^2$ appartenant à deux groupes différents sera un élément de l'ensemble $\mathcal{U}$, et chaque variable $x \in \mathcal{X}$ sera associée à un sous-ensemble couvrant l'ensemble des paires $\{e, e'\}$ pour lesquels une variable x n'a pas la même interprétation par e et e'. Plus formellement, nous définissons pour une variable $x \in \mathcal{X}$ une variable et deux groupes $g, g' \in \mathcal{G}$:

$$\begin{aligned} \mathcal{U} &= \{\{e, e'\} \mid g \neq g', e \in g, e' \in g'\} \\ c_x &= \{\{e, e'\} \mid \{e, e'\} \in \mathcal{U}, e(x) \neq e'(x)\} \\ \mathcal{C} &= \{c_x \mid x \in \mathcal{X}\} \end{aligned}$$

et construisons l'instance $\langle (\mathcal{U}, \mathcal{C}), k \rangle$ de SET-COVER paramétré.

Soit π, une projection de dimension k, solution de d'une instance $\mathcal{I} \equiv (\mathcal{X}, \mathcal{E}, \mathcal{G})$, et $\mathcal{X}' \subseteq \mathcal{X}$ l'ensemble de variables correspondant. Puisque la projection de $\mathcal{I}$ sur $\mathcal{X}$ est satisfiable, pour toute paire d'entité $\{e, e'\} \in \mathcal{E}^2$ appartenant à des groupes différents, on a $e(x) \neq e'(x)$ pour au moins une variable $x \in \mathcal{X}$.

Par définition de c_x, il s'ensuit que la paire $\{e, e'\}$ est dans l'ensemble c_x, et donc, que toute telle paire $\{e, e'\}$ est dans c_x pour au moins un $x \in \mathcal{X}$. Alors la sous-collection $\mathcal{C}' = \bigcup_{x \in \mathcal{X}} c_x$ est bien une solution de l'instance $\langle (\mathcal{U}, \mathcal{C}), k \rangle$ de SET-COVER paramétré.

Réciproquement, soit $C' = \bigcup_{x \in \mathcal{X}}$ une solution de $\langle (\mathcal{U}, \mathcal{C}), k \rangle$, et notons π la projection de $\mathcal{I}$ sur $\mathcal{X}$. Alors par construction, pour toutes instances, si e et e' appartiennent à des groupes différents, il existe une variable $x \in \mathcal{X}$ telle que $\{e, e'\}$ est dans c_x, et donc $e(x) \neq e'(x)$. La projection de $\mathcal{I}$ sur $\mathcal{X}$ est donc satisfiable, ce qui implique que $\mathcal{I}$ est bien une instance positive de PCM paramétré. La réduction est bien une réduction paramétrée, avec accroissement quadratique de l'instance et préservation du paramètre.

Montrons maintenant SET-COVER $\preccurlyeq_{fpt}$ PCM.

Soit $\langle (\mathcal{U}, \mathcal{C}), k \rangle$ une instance de SET-COVER paramétré. Nous construisons une instance $\langle (\mathcal{X}, \mathcal{E}, \mathcal{G}), k \rangle$ de PCM paramétré contenant exactement deux groupes. Le premier groupe g est défini comme contenant une seule entité, notée e_0, et le groupe g' est défini comme contenant une entité e_s pour chaque élément

s de l'ensemble $\mathcal{U}$. On introduit d'autre part une variable x_c par sous-ensemble c dans la collection $\mathcal{C}$. Enfin, nous définissons les valeurs des entités de sorte que l'entité e_s se distingue de l'entité e_0 via la variable x_c si et seulement si l'élément s est dans le sous-ensemble c. Précisément, nous définissons une instance de PCM paramétré par

$$\begin{aligned}
\mathcal{X} &= \{x_c \mid c \in \mathcal{C}\} \\
e_0(x_c) &= 0 \quad \forall x_c \in \mathcal{X} \\
e_s(x_c) &= 1 \quad si\ s \in c \\
e_s(x_c) &= 0 \quad si\ s \notin c \\
\mathcal{E} &= \{e_0\} \cup \{e_s \mid s \in \mathcal{U}\} \\
g &= \{e_0\} \\
g' &= \{e_s \mid s \in \mathcal{U}\} \\
\mathcal{G} &= \{g, g'\}
\end{aligned}$$

La construction est duale de la construction précédente, et il est donc facile de voir que l'instance ainsi construite est une instance de PCM paramétré (de taille k) si et seulement si $\langle(\mathcal{U}, \mathcal{C}), k\rangle$ est une instance positive de SET-COVER paramétré, et que la réduction est bien une réduction paramétrée. □

Corollaire 15 *PCM-paramétré est W[2]-complet.*

La difficulté tient même si l'on se restreint aux instances contenant seulement deux groupes (voir preuve de la proposition 14). Ce résultat nous permet de nous donner une idée plus précise sur la difficulté de résoudre une instance du PCM.

Si nous nous intéressons plus particulièrement à une approche de résolution du problème par niveau, la complexité au pire des cas pour un algorithme de branchement est en $\mathcal{O}(\binom{n}{k})$. Le passage du niveau k à $k+1$ entraîne donc un effort calculatoire conséquent, et la taille de l'instance a un impact direct sur la résolution par rapport à l'explosion faiblement exponentielle confinée au paramètre si caractéristique des problèmes de la classe FPT. Afin d'obtenir une méthode de recherche complète efficace pour le PCM, la seule possibilité d'amélioration significative est de travailler sur les heuristiques de choix de variables.

4.2 Monotonie et langages d'extension

Nous considérons ici la classe des formules positives ne comportant que des littéraux positifs, plus particulièrement les formules en forme normale disjonctive (DNF). Nous montrons alors qu'il est toujours possible de connaître l'existence de telles formules comme solutions d'un PCM en un temps polynomial. Il est également possible de réduire l'espace de recherche sur les variables $\mathcal{X}$, sans pour autant diminuer la classe de complexité du problème.

Nous définissons un ordre sur les valeurs booléennes pour définir par extension un ordre partiel sur les entités.

Définition 16 (ordre $\preceq$ sur {0,1}) *Soit la relation d'ordre $\preceq$ sur {0,1} définie par :*

$$0 \preceq 0 \quad 0 \preceq 1 \quad 1 \preceq 1$$

Définition 17 (Extension de $\preceq$ sur les entités) *L'ordre partiel $\trianglelefteq$ est l'extension aux n-uplets (composante par composante) de l'ordre $\preceq$ sur l'ensemble des entités $\mathcal{E}$.*

Nous nous intéressons maintenant à la satisfiabilité des formules positives par rapport aux entités. Nous montrons que la propriété de satisfiabilité d'une formule positive ϕ_+ est acquise pour toute entité supérieure à une entité satisfaite par ϕ_+. Cette propriété est alors conservée par transitivité sur l'ordre $\trianglelefteq$.

Proposition 18 (monotonie de la satisfiabilité par $\trianglelefteq$ pour des formules positives) *Soient e et e′, deux entités et ϕ_+ une formule positive telle que $e \models \phi_+$. Si $e \trianglelefteq e'$ alors $e' \models \phi_+$.*

Proof *Démonstration par récurrence sur la taille de ϕ_+, une formule positive.*

1. *cas de base où $|\phi_+| = 1$:*
   La seule possibilité est que $\phi_+ = x$ avec $x \in \mathcal{X}$. Or si $e \models \phi_+$ nous avons $e(\phi_+) = 1$ et nous pouvons déduire que $e(x) = 1$. De plus si $e \trianglelefteq e'$ par la définition de l'ordre nous déduisons forcément et uniquement que $e'(x) = 1$, donc $e' \models \phi_+$.
2. *cas de récurrence avec $|\phi_+| = n$, avec $n > 1$:*
   L'hypothèse de récurrence est que si $e \models \phi_+$ et $e \trianglelefteq e'$ alors nous déduisons que $e' \models \phi_+$. Démontrons que pour toute autre formule positive $|\psi_+| = n + 1$ avec $e \models \psi_+$, nous avons $e' \models \psi_+$.
   Nous distinguons 2 cas possibles en considérant la commutativité des opérateurs logiques ET ($\wedge$) et OU ($\vee$).
   - *cas de récurrence avec $\psi_+ = \phi_+ \vee x$ avec $x \in \mathcal{X}$: Nous avons par hypothèse que $e' \models \phi_+$ et par le fait que $\forall e'(x), e' \models \phi_+ \vee x$ nous concluons $e' \models \psi_+$*
   - *cas de récurrence avec $\psi_+ = \phi_+ \wedge x$ avec $x \in \mathcal{X}$: Pour que $e' \models \phi_+ \wedge x$, montrons que $e' \models x$ car par hypothèse $e' \models \phi_+$. Or si $e \models \phi_+ \wedge x$ nous pouvons déduire que $e \models \phi_+$ et que $e \models x$ ce qui implique $e(x) = 1$ Par définition de l'ordre $\trianglelefteq$ et nous déduisons bien que $e'(x) = 1$ donc $e' \models x$.*

□

Le calcul des entités minimales et maximales, par rapport à l'ordre $\trianglelefteq$, de chaque classe C^+ de la pdBf associée à un groupe permet de connaître l'existence d'une projection valide dont les extensions admettent des formules positives. Nous devons montrer qu'aucune entité minimale de la classe C^+n'est supérieure à une entité de la classe C^- pour chaque pdBf de notre problème.

Une deuxième conséquence est que la borne minimale du problème n'est plus égale à $\lceil log_2(|\mathcal{G}|\rceil)$ mais à $|\mathcal{G}|$ avec le test d'inconsistance en $\mathcal{O}(\mathcal{E}^2)$. De plus, la restriction du PCM ayant pour solution un ensemble de formules positives ne nous amène pas à reconsidérer les résultats sur la complexité (la construction de la proposition 14 fournit une instance ayant toujours une extension monotone).

5 Reformulation du PCM en programmation linéaire

Nous présentons maintenant une reformulation du PCM en programmation linéaire. Nous nous intéressons plus particulièrement à la minimisation du PCM (MIN-PCM) qui consiste à minimiser la taille k de la solution. Cette reformulation nous permet d'obtenir de nouveaux résultats de complexité pour le PCM.

5.1 Reformulation

Nous présentons une modélisation du MIN-PCM en programmation entière 0/1 (pseudo-booléen). Nous reformulons le MIN-PCM sous forme d'un problème MIN-ONES. De manière similaire au processus de résolution basé sur les pdBfs, nous nous intéressons aux choix des variables de la solution. Le problème MIN-ONES est un problème d'optimisation défini comme suit.

Définition 19 (MIN-ONES) *Soit Φ une collection de formules booléennes ϕ_i (contraintes) définie sur $\mathcal{X}$. Le problème consiste à trouver une affectation booléenne sur $\mathcal{X}$ telle que chaque contrainte soit satisfaite tout en minimisant le nombre de variable vraie dans cette affectation.*

La réduction vers MIN-ONES reprend l'idée exposée dans la proposition 14. Nous construisons une formule booléenne avec chaque paire d'entités de groupes différents.

Soit $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ une instance du PCM. Pour toute paire d'entités $\{e_i, e_j\} \in \mathcal{E}^2$ telle que $e \in g, e' \in g', g \neq g'$, nous construisons une formule ϕ de la manière suivante :
$\phi = \bigvee_{x_k \in \{x | x \in \mathcal{X}, e(x) \neq e'(x)\}} x_k$.

Nous observons que ϕ est une clause composée uniquement de littéraux positifs. L'ensemble Φ de ces formules permet de modéliser entièrement le PCM de manière à le résoudre par programmation entière 0/1.

$$
\begin{array}{l}
Domaine : y_i \in \{0, 1\} \\
min : \sum_i y_i \\
s.t. \\
y_1 + \cdots + y_n \geq 1, \forall (x_1 \vee \cdots \vee x_n) \in \Phi
\end{array}
$$

Nous constatons que la taille de l'instance résultante est bornée par $|\mathcal{E}|^2$.

Exemple 20 (Transformation vers MIN-ONES en programmation linéaire) *Considérons l'instance :*

Entités	*Groupes*	*Caractères*			
		x1	*x2*	*x3*	*x4*
e1	*g1*	*1*	*1*	*1*	*0*
e2	*g1*	*1*	*1*	*1*	*1*
e3	*g2*	*0*	*0*	*1*	*0*
e4	*g2*	*0*	*1*	*1*	*1*
e5	*g3*	*1*	*1*	*0*	*0*

La transformation en programmation entière 0/1 est la suivante : Pour les contraintes unaires ci-dessous sur y_1 et sur y_3, je changerais $=$ en $\geq$.

$$
\begin{array}{ll}
min : \sum_{i=1}^{4} y_i & \\
s.t. & \\
y_1 + y_2 \geq 1 & c_{\{e1,e3\}} \\
y_1 + y_4 \geq 1 & c_{\{e1,e4\}} \\
y_3 = 1 & c_{\{e1,e5\}} \\
y_1 + y_2 + y_4 \geq 1 & c_{\{e2,e3\}} \\
y_1 = 1 & c_{\{e2,e4\}} \\
y_3 + y_4 \geq 1 & c_{\{e2,e5\}} \\
y_1 + y_2 + y_3 \geq 1 & c_{\{e3,e5\}} \\
y_1 + y_3 + y_4 \geq 1 & c_{\{e4,e5\}}
\end{array}
$$

où chaque contrainte ($c_{\{e_i,e_j\}}$) correspond au traitement d'une paire d'entités.

Avec la transformation proposée, nous pouvons constater qu'une même contrainte ne peut apparaître plusieurs fois, quel que soit le nombre de groupes de départ, si l'instance est d'une part non redondante pour les entités, et d'autre part satisfiable. Bien évidemment cela impose de construire l'ensemble des contraintes en imposant un ordre sur les entités.

Dans [5], les auteurs établissent que MIN-ONES en présence de clauses ne possédant que des littéraux positifs est contenu dans la classe de complexité APX [22], ce qui implique que PCM est approximable.

5.2 Réduction d'une instance du problème en programmation linéaire

La transformation du PCM afin de le résoudre à l'aide de programmation linéaire produit un nombre extrêmement important de clauses. Afin de réduire ce nombre, il est envisageable d'utiliser la détection de subsomptions [6], comme pour le problème SAT. Ce mécanisme peut s'appliquer à la simplification d'une instance en programmation entière 0/1 car la définition particulière des contraintes sur lesquelles nous travaillons le permet (clauses ne contenant que des littéraux positifs). Si $var(c_{\{e_i,e_j\}})$, avec $i \neq j$, désigne l'ensemble des variables de décision d'une contrainte produite par la paire d'entités $\{e_i, e_j\}$,

nous pouvons supprimer toute contrainte ayant un sous-ensemble de variables correspondant à l'ensemble de variables d'une contrainte. Plus formellement, nous supprimons toute contrainte $c_{\{e_i,e_j\}}$ telle que pour un $c_{\{e_k,e_l\}}$, avec $k \neq l$, nous ayons $var(c_{\{e_k,e_l\}}) \subset var(c_{\{e_i,e_j\}})$. Comme pour la subsomption, il est facile de remarquer que n'importe quelle affectation validant la contrainte $c_{\{e_k,e_l\}}$ implique la validité de $c_{\{e_i,e_j\}}$.

Dans l'exemple précédent, nous calculons la solution $y_1 = 1$ et $y_3 = 1$ (les autres à 0) en remarquant que $c_{\{e2,e4\}}$ (resp. $c_{\{e1,e5\}}$) permet de supprimer $c_{\{e1,e3\}}$, $c_{\{e1,e4\}}$, $c_{\{e2,e3\}}$, $c_{\{e3,e5\}}$ et $c_{\{e4,e5\}}$ (resp. $c_{\{e2,e5\}}$).

Je trouve la suite pas claire, ni les items ni le calcul de probas. Toutefois, il faut remarquer qu'une subsomption n'est présente que lorsque les conditions suivantes sont vérifiées :

- deux entités e_1 et e_2 d'un même groupe possèdent x variables avec la même valeur ;
- au moins une entité d'un autre groupe a des valeurs identiques pour les $n - x$ variables non identiques de e_1 ou de e_2 (n étant le nombre total de variables).

La probabilité de réduire une instance en programmation linéaire est donc assez faible ($\frac{1}{2^{2n}} < p < \frac{1}{2^n}$), mais nous verrons dans la partie expérimentale que pour des instances venant de problèmes réels, les conditions de subsomption sont souvent réunies.

6 Application : test de diagnostic en biologie végétale

Nous présentons dans cette section l'utilisation du PCM dans le contexte de la biologie végétale.

6.1 Caractérisation de pathovars

Comme mentionné en introduction, la bactérie *Xanthomonas* cause de gros dégâts sur certaines cultures. Les différentes souches de cette bactérie sont réparties en groupes et leurs caractéristiques (génotypiques ou phénotypiques) sont connues. L'objectif est de créer un test de diagnostic permettant de détecter rapidement à quel groupe appartient une souche donnée.

La caractérisation précise des collections de souches bactériennes est un enjeu scientifique majeur, car ces bactéries sont responsables d'importantes pathologies végétales, et donc soumises à des procédures de contrôle officielles (par exemple, en Europe, la directive 2000/29/CE). Le développement de tests de diagnostic est donc crucial pour identifier systématiquement les souches de ces espèces. Dans ce contexte, le problème de 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 [14]. Une souche peut donc être vue comme un vecteur de valeurs binaires qui reflète la présence (valeur 1) ou

l'absence (valeur 0) de ces caractères. De plus, les tests de diagnostic étant basés sur des puces à ADN, il est nécessaire de chercher à minimiser les solutions (le nombre de tests associés pour chaque identification) afin de réduire le coût prohibitif de ces tests. Notons également que pour chaque caractère le spot de test doit être dupliqué sur la puce, ce qui réduit d'autant plus le nombre de caractères utilisables pour un même kit de diagnostic. Ce problème correspond exactement à MIN-PCM.

6.2 Expérimentations

Nous avons mené des expérimentations sur des instances réelles et des instances générées aléatoirement.

- Les instances réelles (raphv, raphy, rarep, rch8 et rch10) sont tirées de problèmes de caractérisation provenant de l'API BioMérieux, basée sur des propriétés bio-chimiques de l'espèce Ralstonia et sur des expressions de gènes de virulence de l'espèce Xanthomonas.
- Les instances aléatoires sont générées en deux étapes, pour un nombre de variables et de groupes fixés. Tout d'abord, chaque entité est construite aléatoirement. Ensuite, une certaine proportion des entités est répartie de manière équitable entre tous les groupes, et les entités restantes sont affectées une à une aléatoirement aux groupes. La proportion d'entités affectées aléatoirement est donnée en pourcentage et est appelée *bruit*. Une instance aléatoire est notée sous la forme : s*graine-bruit*.

La Table 1, présente plusieurs informations sur les instances ainsi que sur leur résolution par différents algorithmes sont présentées. Les colonnes « entités », « groupes » et « caractères » donnent les propriétés des instances. La colonne « contraintes » fournit le nombre de contraintes nécessaires pour reformuler chaque instance sous forme de programme linéaire, la colonne r désigne le ratio entre le nombre de contraintes et le nombre d'entités, et la colonne réduction indique le nombre de contraintes après réduction (« non » si aucune). Les trois dernières colonnes fournissent les résultats obtenus (nombre de variables dans la caractérisation trouvée) pour la résolution en programmation linéaire en utilisant *cplex*[2] sur le problème réduit(« PL »), pour l'exécution d'*Exact-Proj-Car* (« EPJ »), et enfin pour l'application de l'algorithme de recherche locale *LS-Proj-Car* (« LSPC »). Le temps autorisé pour chaque exécution est limité à 10 minutes. Les résultats en gras indiquent que les solutions sont optimales.

Nous constatons que le nombre de contraintes est beaucoup plus important que le nombre d'entités de départ, mais il reste faible par rapport à la borne théorique exprimée plus haut. Comme mentionné dans la section 5.2, la réduction pour les instances aléatoires n'a aucun effet, car les conditions nécessaires n'ont qu'une très faible probabilité d'être vérifiées. Toutefois, il est intéressant d'observer que pour les instances réelles, le taux de réduction est assez élevé, ce qui est cohérent car les groupes sont constitués d'entités très similaires. Le prétraitement nécessaire à la reformulation du problème sous forme de pro-

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

Instances	Entités	Groupes	Caractères	Contraintes	r	Réduction	PL	EPC	LSPC
s301-0	500	30	400	120640	241	non	-	13	14
s326-0	500	10	500	112500	225	non	-	13	14
s413-30	500	20	600	114358	229	non	-	13	14
s555-20	800	20	800	113420	227	non	-	13	14
s625-20	500	5	10000	99485	199	non	-	13	14
s754-10	600	10	200	159728	266	non	-	13	14
s882-20	600	10	400	158871	265	non	-	13	14
s2501-70	800	10	800	276232	345	non	-	14	15
s31294-50	200	15	10000	16823	84	non	10		11
s3836-0	1000	15	10000	475000	475	non	-	16	
raphv	109	8	155	4091	37	2200	**6**		9
raphy	113	4	155	3435	30	1887	**6**		8
rarep	112	7	155	4587	41	97	**12**	59	13
rch8	132	21	37	1631	12	58	**9**		9
rch10	173	27	98	5908	34	983	**10**	15	14

TABLE 1 – Résultats expérimentaux

grammation linéaire peut nécessiter de quelques secondes pour les plus petites instances à plusieurs minutes pour les plus grosses.

Étant limités en mémoire (32Go sur une machine 64 bits), nous n'avons pu lire (chargé en mémoire) les instances aléatoires (noté par « - ») avec le solveur *cplex*. Nous remarquons que pour l'instance s31294, cette instance est lu par *cplex* mais malheureusement nous n'avons pu résoudre le problème de manière optimale en raison également de la mémoire insuffisante. Pour les problèmes réels *cplex* est rapide (quelques secondes) et optimal. Il est d'ailleurs bien plus rapide que le solveur *Exact-Proj-Car* (de l'ordre de la minute). Cependant *Exact-Proj-Car* fournit une borne supérieure en utilisant très peu de mémoire (environ 20 Mo) pour les instances aléatoires. *LS-Proj-Car* permet de garantir en toute circonstance une borne optimale facilement calculable avec peu de mémoire et avec une exécution de l'ordre de la seconde, mais il n'obtient pas jamais la solution optimale.

Les méthodes de résolution que nous avons proposées permettent de traiter des problèmes réels de taille conséquente. Des caractérisations dans le domaine de la biologie végétale ont donc été réalisées, et ont conduit au dépôt d'un brevet pour le dépistage du pathovar phaseoli de Xanthomonas Axonopodis [3]. Ce test vient d'être validé au niveau européen comme un des tests officiels de dépistage de ce pathovar. Nous avons de plus mis à disposition une page web [3] permettant aux biologistes d'obtenir aisément une caractérisation de données correspondant au PCM.

3. `http://forge.info.univ-angers.fr/~gh/Idas/Ccd/mcps/`

7 Conclusion

Dans cet article, nous avons défini formellement le problème de caractérisation multiple (PCM), qui consiste à trouver une formule booléenne qui caractérise chaque groupe d'entités représenté par un ensemble d'interprétations booléennes. Nous avons proposé deux méthodes exactes et une méthode de recherche locale afin de le résoudre.

Une étude de la complexité de ce problème a permis de montrer qu'il n'existait pas d'algorithme FTP (fixed parameter tractable) pour le résoudre. Une réduction vers le problème SET-COVER amène en effet à la conclusion que le PCM est W[2]-Complet. Cette complexité permet de savoir que les informations apprises lors de la recherche d'une caractérisation de taille k par une méthode exacte sont difficilement utilisables pour la recherche d'une caractérisation de taille $k-1$ ou $k+1$.

Nous avons proposé un codage du problème en programmation linéaire à l'aide d'une transformation vers le problème MIN-ONES. Ceci nous a permis de comparer différentes méthodes de résolutions (exacte, recherche locale, programmation linéaire) pour des instances aléatoires et réelles du PCM. Les instances aléatoires semblent plus difficiles à résoudre que les instances réelles, et un travail futur sera d'étudier la génération d'instances afin de définir ce qui rend une instance difficile (nombre d'entités, nombre de groupes, diversité intra et intergroupe). Pour les instances réelles, de bons résultats ont été obtenus. Ils ont été valorisés par le dépôt d'un brevet pour le dépistage du pathovar phaseoli de Xanthomonas Axonopodis ainsi que par une validation au niveau européen comme un des tests officiels de dépistage de ce pathovar.

Références

[1] James Bailey, Thomas Manoukian, and Kotagiri Ramamohanarao. Fast algorithms for mining emerging patterns. In *Principles of Data Mining and Knowledge Discovery*, pages 39–50. Springer, 2002.

[2] T. Boureau, M. Kerkoud, F. Chhel, G. Hunault, A. Darrasse, C. Brin, K. Durand, A. Hajri, S. Poussier, C. Manceau, F. Lardeux, F. Saubion, and M.A. Jacques. A multiplex-pcr assay for identification of the quarantine plant pathogen xanthomonas axonopodis pv. phaseoli. *Journal of Microbiological Methods*, 92(1) :42–50, 2013.

[3] Tristan Boureau, Fabien Chhel, Gilles Hunault, Mohamed Kerkoud, Frédéric Lardeux, Charles Manceau, Stéphane Poussier, and Frédéric Saubion. Procédé de depistage de xanthomonas axonopodis pv. phaseoli. Brevet INRA Depose INPI : FR 2970480 (A1), 2012.

[4] Fabien Chhel, Frédéric Lardeux, Adrien Goëffon, and Frédéric Saubion. Minimum multiple characterization of biological data using partially defined boolean formulas. In *SAC*, pages 1399–1405, 2012.

[5] N. Creignou, N.C.S.K. Madhu Sudan, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Siam Monographs on Discrete Mathematics and Applications. 2001.

[6] S. Darras, G. Dequen, L. Devendeville, B. Mazure, R. Ostrowski, and L. Saïs. Using boolean constraint propagation for sub-clauses deduction. In *Principles and Practice of Constraint Programming - CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 757–761. Springer, 2005.

[7] P. Descamps and M. Véon. Une méthode de choix des caractères d'identification basée sur le théorème de bayes et la mesure de l'information. *Ann. Microbiol. (Paris)*, 132B, 1981.

[8] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness i : Basic results. *SIAM J. Comput.*, 24(4) :873–921, 1995.

[9] Rodney G. Downey and Michael R. Fellows. *Parameterized complexity*. Monographs in Computer Science, Springer-Verlag, 1999.

[10] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[11] Ricard Gavaldà and Denis Thérien. An algebraic perspective on boolean function learning. In *Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings*, volume 5809 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 2009.

[12] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[13] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3 :1157–1182, 2003.

[14] Ahmed Hajri, Chrystelle Brin, Gilles Hunault, Frédéric Lardeux, Christophe Lemaire, Charles Manceau, Tristan Boureau, and Stéphane Poussier. A "repertoire for repertoire" hypothesis : Repertoires of type three effectors are candidate determinants of host specificity in xanthomonas. *PLoS ONE*, 4(8) :e6632, 08 2009.

[15] Lisa Hellerstein and Rocco A. Servedio. On pac learning algorithms for rich boolean function classes. *Theor. Comput. Sci.*, 384(1) :66–76, 2007.

[16] Holger Hoos and Thomas Stützle. *Stochastic Local Search : Foundations & Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[17] Kazuhisa Makino, Ken ichi Hatanaka, and Toshihide Ibaraki. Horn extensions of a partially defined boolean function. *SIAM J. Comput.*, 28(6) :2168–2186, 1999.

[18] E.J. McCluskey. Minimization of boolean functions. *Bell System Tech. J*, 35 :1417–1444, 1956.

[19] David Meyer, Friedrich Leisch, and Kurt Hornik. The support vector machine under test. *Neurocomputing*, 55(1-2) :169–186, 2003.

[20] Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning juntas. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC 2003)*, pages 206–212, 2003.

[21] B. K. Natarajan. On learning boolean functions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 296–304. ACM, 1987.

[22] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3) :425–440, 1991.

[23] Ruggero G. Pensa, Céline Robardet, and Jean-François Boulicaut. Supporting bi-cluster interpretation in 0/1 data by means of local patterns. *Intell. Data Anal.*, 10(5) :457–472, 2006.

[24] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5 :239–266, 1990.

[25] N.W. Schaad, E. Postnikova, G.H. Lacy, A. Sechler, I. Agarkova, P.E. Stromberg, V.K. Stromberg, and A.K. Vidaver. Reclassification of xanthomonas campestris pv. citri (ex hasse 1915) dye 1978 forms a, b/c/d, and e as x. smithii subsp. citri (ex hasse) sp. nov. nom. rev. comb. nov., x. fuscans subsp. aurantifolii (ex gabriel 1989) sp. nov. nom. rev. comb. nov., and x. alfalfae subsp. citrumelo (ex riker and jones) gabriel et al., 1989 sp. nov. nom. rev. comb. nov. ; x. campestris pv malvacearum (ex smith 1901) dye 1978 as x. smithii subsp. smithii nov. comb. nov. nom. nov. ; x. campestris pv. alfalfae (ex riker and jones, 1935) dye 1978 as x. alfalfae subsp. alfalfae (ex riker et al., 1935) sp. nov. nom. rev. ; and "var. fuscans" of x. campestris pv. phaseoli (ex smith, 1987) dye 1978 as x. fuscans subsp. fuscans sp. nov. *Syst Appl Microbiol.*, 28(6) :494–518, 2005.

[26] Christopher Umans. The minimum equivalent DNF problem and shortest implicants. *J. Comput. Syst. Sci.*, 63(4) :597–611, 2001.

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

[28] L. Vauterin, B. Hoste, K. Kersters, and J. Swings. Reclassification of xanthomonas. *Int. J. Syst. Bacteriol.*, 45 :472–489, 1995.