HAL
open science

Problème de caractérisation multiple : Application à la détection de souches bactériennes phytopathogènes

Fabien-Sothéa Chhel

To cite this version:

Fabien-Sothéa Chhel. Problème de caractérisation multiple : Application à la détection de souches bactériennes phytopathogènes. Informatique et langage [cs.CL]. Université d'Angers, 2014. Français. NNT : 2014ANGE0018 . tel-01531194

HAL Id: tel-01531194

https://theses.hal.science/tel-01531194

Submitted on 1 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse de Doctorat

Fabien-Sothéa
CHHEL

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université d'Angers
sous le label de l'Université de Nantes Angers Le Mans

École doctorale : Sciences et Technologies de l'Information, et Mathématiques (STIM)

Discipline : Informatique et applications, section CNU 27
Unité de recherche : Laboratoire d'Études et de Recherches en Informatique d'Angers (LERIA)

Soutenue le 1 octobre 2014
Thèse n° : 1372

Problème de caractérisation multiple

Application à la détection de souches bactériennes phytopathogènes

JURY

Rapporteurs :	**M. Gilles AUDEMARD**, Professeur, Université d'Artois
	M. Simon DE GIVRY, Chargé de recherche - HDR, INRA - Toulouse
Examinateurs :	**Mme Clarisse DHAENENS**, Professeur, Université Lille 1
	M. Arnaud LALLOUET, Professeur, Université de Caen
Directeur de thèse :	**M. Frédéric SAUBION**, Professeur, Université d'Angers
Co-directeur de thèse :	**M. Frédéric LARDEUX**, Maître de conférences - HDR, Université d'Angers

Table des matières

Introduction générale **9**

I Contexte du problème de caractérisation multiple **15**

1 Contexte de l'identification d'une souche bactérienne **17**
1.1 Vers une utilisation croissante de l'informatique en biologie . 18
1.2 Identification de pathovars 20
1.3 Analyse des souches par PCR 22
1.4 Conclusion 25

2 Logique propositionnelle, optimisation combinatoire et complexité **27**
2.1 Logique propositionnelle 29
2.1.1 Syntaxe 29
2.1.2 Sémantique 31
2.1.3 Formes normales 33
2.1.4 Fonctions booléennes 36
2.2 Optimisation combinatoire 39
2.2.1 Problèmes d'optimisation combinatoire 39
2.2.2 Panorama des méthodes de résolution 41
2.2.3 Discussions 49
2.3 Quelques notions de complexité 50
2.3.1 Classes de complexité 51
2.3.2 Réductions 54
2.3.3 Autres classes de complexité 56
2.4 Conclusion 60

3 Quelques approches pour l'apprentissage et la construction de formules booléennes **61**
3.1 Apprentissage automatique 63
3.2 Apprentissage supervisé 64
3.3 Sélection d'attributs 65
3.3.1 Méthodes de régression 67
3.3.2 SVM 68
3.4 Minimisation de formule DNF 70
3.5 Conclusion 71

II Contribution au problème de caractérisation multiple **73**

4 Formalisation du problème de caractérisation multiples (PCM) **75**
4.1 Contexte du PCM 76
4.2 Définitions et propositions 78
4.3 Résolution par les fonctions booléennes partiellement définies (pdBfs) 82
4.4 NP-Complétude et FPT-Hiérarchie 84
4.5 Autres propriétés 87
4.5.1 Bornes et satisfiabilité 87
4.5.2 Monotonie et langages d'extension 87
4.6 Conclusion 89

5 Présentation des algorithmes **91**
5.1 Approche par générations de formules 93
5.1.1 Quilan au premier ordre Prolog - EXACTFORMCAR 93
5.1.2 Algorithme génétique 95
5.1.3 Limites de ces approches « directes » 97
5.2 Approche par les pdBfs 97
5.2.1 Heuristique et voisinage de recherche 97
5.2.2 Recherche complète 98
5.2.3 Recherche locale 99
5.3 Reformulation du PCM en programmation linéaire 99
5.3.1 Règles de recodage 100
5.3.2 Simplifications d'instance 101
5.4 Conclusion 102

6 Expérimentations et valorisation **103**
6.1 Instances 104
6.1.1 Instances Réelles 104
6.1.2 Instances Aléatoires 104
6.2 Travaux préliminaires et problèmes connexes 105
6.2.1 Coefficient de Capacité Diagnostique (CCD) 105
6.2.2 Caractérisations Positives et Canonicité 106
6.2.3 Caractérisations groupe par groupe 106
6.3 Expérimentations 107
6.3.1 Valorisation 109
6.4 Conclusion 109

Conclusion générale **111**

Annexes **113**

Contributions personnelles **119**

Références bibliographiques **121**

Remerciements

La thèse est une aventure intellectuelle mais elle est aussi le fruit des rencontres, des discussions de débats et aussi de détente.

Mes premiers remerciements sont destinés à mes encadrants Frédéric Saubion et Frédéric Lardeux pour leur confiance accordée et leur accompagnement durant ces années. Ils ont été présents à chaque moment de ce travail et j'ai pu partager de nombreux moments.

Je voudrais remercier mes coauteurs. Tristan Boureau qui sans lui, je n'aurais pu commencer ce travail en sachant nous exposer les problèmes de diagnostiques de pathovars. Gilles Hunault pour ces remarques pertinentes et sa curiosité qui a su attiser la mienne. Adrien Goëffon pour son expertise sur les métaheuristiques. Bruno Zanuttini qui a su comprendre les aspects théoriques abordés par cette thèse et a su me guider pour apporter les arguments formels. Mon ami Adrien Salomé pour avoir participé à la résolution d'un problème d'optimisation durant sa thèse en physique optique.

Je tiens aussi à remercier tous les membres de ce jury de thèse. Gilles Audemard et Simon de Givry pour leur travail de rapporteur et qui ont enrichi ce manuscrit par leurs questions et leurs remarques. Clarisse Dhaenens qui a accepté d'être présidente et Arnaud Lallouet pour son rôle d'examinateur.

Mes remerciements suivants sont destinés aux personnes avec qui j'ai partagé mon travail quotidiennement. Je remercie tout le personnel du département d'informatique et du LERIA. Les enseignants-chercheurs dont la majorité a été mes professeurs, les techniciens et ingénieurs pour l'excellent support du matériel, les administratifs qui nous ont accompagnés dans nos démarches et bien sûr tous mes collègues doctorants et post-doctorants dont une pensée spéciale à Julien Robet.

Enfin, je voudrais terminer ces remerciements par Nadarajeen Veerapen avec qui je me suis lié d'amitié. J'ai apprécié sa discrétion et sa gentillesse. Je n'oublie pas non plus Giacomo di Tollo. Nous avons passé de merveilleux moments dans notre bureau et notre collocation. Un grand merci à Coralie Auder pour la relecture de ce manuscrit. Et bien sûr merci à ma famille, à tous mes amis d'Angers et d'ailleurs.

Introduction générale

Contexte général

Selon le dictionnaire Larousse, l'informatique est « la science du traitement automatique et rationnel de l'information considérée comme le support des connaissances et des communications ». Cette définition laisse entrevoir la capacité de nos chers ordinateurs via leurs programmes à acquérir des informations sous forme de données, de les organiser et les traiter en vue de fournir des solutions pour les problèmes spécifiés. En anglais, l'informatique correspond au terme « computer science » avec une traduction littérale se rapprochant de la « science du calcul ». Cette thèse aborde ces deux aspects de l'informatique puisque d'une part nous nous intéressons à un traitement de données biologiques sous la modélisation de la logique propositionnelle et d'autre part nous essayons de comprendre et de déterminer les conséquences en terme de complexité et donc de temps de résolution.

Avec l'arrivée de nouvelles technologies d'acquisition de données la génomique a pu progresser de manière significative et ainsi des génomes complets sont disponibles pour des espèces vivantes ou végétales. Dans ce contexte, on assiste à une accumulation de données accessibles alors que les outils d'analyse et d'extraction d'informations pertinents font souvent défaut. Á l'instar du problème posé par le déluge d'informations sur les réseaux, le traitement des données biologiques constitue un enjeu socio-économique de tout premier ordre pour les prochaines années. Il apparaît ainsi clairement dans les orientations stratégiques scientifiques de nombreux programmes de recherche à l'échelle mondiale.

De même, que les interactions entre les biologistes et les acteurs des sciences du numérique, des mathématiques, de la physique se sont développées de manière très importante ces dernières années pour fournir à la fois des modèles de représentation mais aussi des outils de calcul performants pour acquérir et traiter les données biologiques. De nouveaux domaines ont alors émergé il y a quelques années, telle la bio-informatique, et sont maintenant clairement identifiés à l'intersection de plusieurs disciplines.

Les possibilités accrues d'acquisition de données génomiques, grâce au séquençage à haut débit, mais également phénotypiques, pose le problème de la caractérisation de données biologiques. Cette dernière devrait rapidement devenir l'un des verrous essentiels de l'exploitation effective des grandes bases de données et constituera donc un centre d'intérêt commun pour les biologistes des domaines du végétal ou de la santé. La caractérisation telle que nous l'entendons dans cette thèse se positionne en aval des approches génomiques et phénotypiques, pour tenter d'identifier les caractères propres, éventuellement hétérogènes, d'un groupe d'individus partageant des spécificités fonctionnelles communes (par exemple pathologiques).

En considérant un sous-ensemble d'individus représentatifs de différents groupes (genre, espèce, souche,...), il s'agit d'établir pour chacun de ces groupes une caractérisation permettant de distinguer (c'est-à-dire discriminer) les individus du groupe par rapport à ceux des autres groupes. De plus, nous souhaitons naturellement minimiser le nombre de caractères qu'il sera nécessaire d'observer pour effectuer cette caractérisation multiple. Nous utilisons le terme « caractérisation multiple » pour rendre compte de l'aspect mutuel de ces discriminations au sens « un groupe versus le reste des groupes » et ce pour chaque groupe.

Problèmes de caractérisation de données biologiques

En se concentrant sur les problèmes de caractérisation tels que nous venons de les évoquer, nous pouvons mettre en avant les différentes phases que nécessite un tel projet scientifique.

1. **Détermination de caractères mesurables sur les individus et définition d'une méthodologie de mesure de ces caractères** (acquisition de données par séquençage, utilisation de puces à ADN, imagerie...). Cette phase peut inclure des caractères purement génomiques ou phénotypiques et ainsi faire appel à des techniques et des appareillages d'acquisition très variés. Cette première phase est clairement prise en charge principalement par les biologistes en interaction, pour certains types de procédés, avec des champs disciplinaires complémentaires tels que la biologie moléculaire pour la mise au point de nouveaux processus d'analyse comme la PCR (réaction en chaîne par polymérase) ou encore la photonique ou l'électronique pour de nouvelles techniques d'imagerie...

2. **Traitement de données brutes**. Les données extraites des mesures précédentes sont souvent incomplètes ou bruitées et nécessitent des traitements préliminaires avant d'être analysées plus finement. De nouveau, ces traitements peuvent nécessiter des compétences larges comme par exemple des méthodes de traitement d'images et du signal ou encore des approches statistiques pour identifier a priori un sous-ensemble de données pertinentes. Ici encore, ces démarches doivent s'effectuer en étroite collaboration avec les biologistes afin de valider la pertinence des hypothèses retenues.

3. **Modélisation**. Une formalisation du problème identifiant clairement ces données, les contraintes auxquelles elles sont soumises et les résultats attendus est nécessaire. Nous utilisons à dessein le vocabulaire de l'optimisation combinatoire puisque c'est dans ce contexte de modélisation de problème que nous allons nous plonger. Il s'agit dans cette étape de fournir, encore une fois de concert avec les utilisateurs finaux, un modèle réaliste mais également aborder le problème d'un point de vue calculatoire (ou même *computationnel* puisque qu'à partir de cette étape, l'informatique prend toute sont importance).

   Le choix d'un paradigme de modélisation est souvent essentiel dans ce type d'approche puisqu'il aura inévitablement un impact sur le choix des méthodes algorithmiques qui seront mises en œuvre pour la résolution du problème à traiter. En général omises dans le cadre de la présentation des résultats, notons que de nombreuses tentatives infructueuses sont souvent nécessaires et utiles à ce stade pour pouvoir être le plus en adéquation avec le modèle réel et faire face à des évolutions naturelles des attentes des utilisateurs biologistes.

4. **Analyse du modèle et proposition d'approches de calculs**. La démarche d'un informaticien (que nous envisageons ici plutôt comme un *computer scientist*) consiste alors naturellement à s'intéresser à la complexité calculatoire du problème ainsi modélisé. En amont, il est pertinent de se poser pour certains problèmes la question même de la calculabilité. Une fois la complexité caractérisée, des méthodes algorithmiques adaptées peuvent alors être envisagées et testées.

5. **Expérimentation sur des données réelles**. Cette phase n'est pas la plus facile car si les données brutes biologiques sont accessibles en grande quantité, la définition de jeux de tests appropriés requiert une sélection de la part des biologistes afin d'extraire des données pertinentes relatives à une problématique spécifique. L'acquisition de

données, décrite plus haut est souvent longue, fastidieuse et coûteuse. Notons par exemple qu'une des caractéristiques intrinsèques à la biologie végétale qui va nous intéresser ici est que la plante pousse à son rythme...

6. **Valorisation des résultats**. Une fois l'obtention d'outils de caractérisation de données, ils doivent être mis en œuvre de manière pratique au sein d'un processus de diagnostic. Dans cette thèse, par exemple, un test de diagnostic de pathologie végétale a ainsi été défini et a nécessité, outre les aspects modélisation et calcul, la mise au point d'une puce à ADN spécifique (et donc de réactifs appropriés). Une fois ce test conçu, il doit être utilisé sur des jeux de données réelles suffisamment représentatifs, s'insérant ainsi dans des procédure de tests et de validation.

Il est dès lors clair que le travail présenté dans cette thèse se concentre sur les phases 3 et 4 que nous venons de décrire, la phase 5 ayant été également abordée avec succès. Toutefois, il convient de remarquer que ce travail ne peut être effectué sans un minimum de compréhension des phases 1 et 2, ce qui implique une ouverture disciplinaire enrichissante.

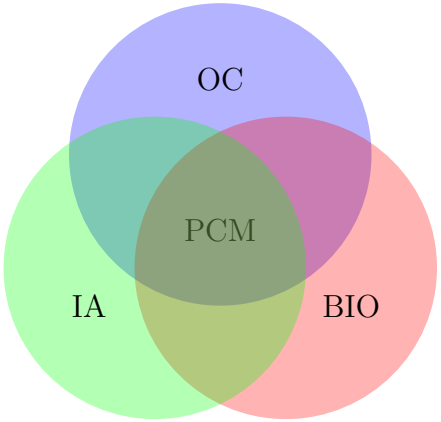


Figure 1 : Digramme de Venn permettant de situer le Problème de Caractérisation Multiple (PCM) en fonction des domaines de l'optimisation combinatoire (OC), de la biologie (BIO) et de l'intelligence artificiel (IA) en particulier l'apprentissage.

Enfin, nous résumons par la figure 1 le contexte disciplinaire de nos recherches.

Contributions

Nous précisons à présent le contexte biologique de notre contribution ainsi que les approches que nous avons développées dans le cadre du problème de caractérisation multiple.

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 suffisent à générer une source d'inoculation primaire et à occasionner ainsi une dissémination ultérieure plus large.

En particulier, le pathovar *phaseoli* de *Xanthomonas axonopodis* (*Xap*) qui regroupe toutes les souches identifiées comme pathogènes sur le haricot 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 ; 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 à *Xap*. Cette combinaison peut ainsi être utilisée pour concevoir un test PCR multiplex pour l'identification de *Xap*. Nous avons ainsi participé au développement d'un test moléculaire pour la détection rapide de toutes les souches de *Xanthomonas* pathogènes sur les haricots [Boureau et al., 2013].

Sur le plan des contributions informatiques nous avons défini le problème de caractérisation multiple comme le problème de construction de formules booléennes propositionnelles étant donné un ensemble de tables de vérités partielles de ces formules. Par rapport aux travaux existants dans ce domaine, nous avons eu à introduire la notion de minimisation du nombre de littéraux utilisés dans cet ensemble de formules. Le problème ainsi défini a été caractérisé en terme de complexité, utilisant des notions de complexité paramétrée. Nous avons également proposé plusieurs approches de résolution du problème incluant un algorithme de recherche exhaustive en arborescence, une approche par programmation linéaire en nombre entier et des algorithmes approchés de type métaheuristiques (recherche locale et algorithme évolutionnaire).

Des expérimentations ont été menées pour mettre en avant la difficulté de résolution et les problèmes de passage à l'échelle pour certaines approches de résolution.

Organisation du manuscrit

La première partie de ce manuscrit présente le problème de caractérisation multiple dans le contexte biologique de l'identification de bactéries phytopathogènes. Nous introduisons au chapitre 1 le vocabulaire nécessaire à la compréhension des enjeux de ce domaine de la biologie et nous présentons succinctement les différents tests de diagnostic actuels ainsi que leurs mises en œuvre. Dans le chapitre 2, nous représentons les outils formels que sont la logique propositionnelle et la complexité algorithmique dans le but de définir et décrire le problème de caractérisation multiple sous l'angle de l'optimisation combinatoire. Nous exposerons aussi l'ensemble des principes de résolutions complètes et incomplètes développés au cours de cette thèse. Nous terminerons cette première partie (chapitre 3) sur les approches présentes dans la littérature et pouvant se rapprocher de notre problématique.

La deuxième partie porte sur nos contributions et sur le développement de nos travaux. Nous proposons conjointement avec les biologistes un ensemble de définitions issues de la modélisation (chapitre 4) qui nous permettent d'énoncer le problème de caractérisation multiple. Nous présentons aussi des aspects plus théoriques qui nous ont aiguillé et ont justifié nos choix sur le développement de nos algorithmes (chapitre 5). Nous terminons (chapitre 6) ce manuscrit par une discussion sur les instances et les résultats obtenus dans le cadre ce thèse.

En annexe se situent quelques informations sur le solveur final produit et son utilisation.

I

Contexte du problème de caractérisation multiple

1

Contexte de l'identification d'une souche bactérienne

Sommaire

1.1 Vers une utilisation croissante de l'informatique en biologie 18
1.2 Identification de pathovars 20
1.3 Analyse des souches par PCR 22
1.4 Conclusion 25

Ce chapitre est consacré à la description du problème de caractérisation multiple d'un point vue biologique. Nous aborderons ainsi les différents concepts biologiques concernant la détection de bactéries phytopathogènes.

1.1 Vers une utilisation croissante de l'informatique en biologie

La bio-informatique est depuis plusieurs années un champ disciplinaire de l'informatique qui subit une très forte croissance. Multidisciplinaire par essence, la bio-informatique réunit ainsi des biologistes, des informaticiens, des statisticiens, des chimistes ou encore des physiciens pour mettre en œuvre la résolution de problèmes importants issus du domaine de la biologie. Les problèmes ainsi traités sont par exemple l'étude du génome, la modélisation de processus biologiques ou bien encore l'évolution des espèces.

En particulier, l'amélioration constante des techniques d'acquisition de données génomiques (séquençage à haut débit) a permis d'enrichir considérablement les banques de données. Ces avancées sont toutefois pondérées par le manque de techniques efficaces pour exploiter pleinement ces données. Après avoir utilisé largement le concept de *big data*, qui concerne globalement l'ensemble des données qui sont actuellement produites, que ce soit individuellement ou par le biais d'organisations (entreprises, laboratoires de recherche, administrations...), le terme de déluge de données (*data deluge*) est même maintenant utilisé, notamment par les biologistes conscient des difficultés que vont entraîner ces nouvelles méthodes d'acquisition. Même s'il est difficile de faire un état des lieux exhaustif des différentes problématiques qui sont abordées à l'heure actuelle dans le cadre de la bio-informatique et, plus généralement, dans le contexte des nombreuses interactions qui existent entre les sciences du numérique et la biologie, nous pouvons citer, à titre d'exemples, les aspects suivants :

- Interprétation de données génomiques : alignement de séquences [Altschul et al., 1990], phylogénie [Nei and Kumar, 2000], classification de données issues de puces à ADN [Simon, 2003],...
- Modélisation de mécanismes biologiques : conformation de protéines [Braun and Gō, 1985], systèmes de régulations géniques [Matsuno et al., 2000],...

Du point de vue des sciences du numérique, les approches utilisées relèvent de champs assez divers tels que le traitement du signal, l'analyse d'images, la classification et l'apprentissage automatique, l'optimisation.

À terme, les perspectives sont nombreuses et revêtent des enjeux sociétaux importants, comme par exemple, la notion de systèmes médicaux personnalisés permettant, en fonction des caractéristiques génomiques mais

également phénotypiques d'un patient, de lui proposer le traitement thérapeutique le plus adapté.

D'autre part dans cette thèse, est aussi explorée la perspective d'identifier clairement les spécificités phénotypiques d'espèces végétales afin de pouvoir améliorer, par exemple, la qualité des semences en fonction de l'environnement géographique et climatique mais aussi des risques présents.

Les travaux présentés dans cette thèse se positionnent dans le cadre de recherches en biologie sur la détection de bactéries phytopathogènes. Il s'agit en particulier d'identifier clairement des groupes de bactéries phytopathogènes qui s'attaquent à différentes plantes, notamment des plantes cultivées pour des raisons alimentaires (fruits et légumes en particulier). Ainsi, ces bactéries qui partagent des caractéristiques génomiques communes peuvent occasionner des effets assez divers en fonction des végétaux qu'elles attaquent (ceci est communément appelé la « spécificité d'hôte »). Leur identification est rendue plus complexe par le fait qu'une même souche bactérienne, ou des souches génétiquement proches, peuvent ainsi présenter des traits phénotypiques différents alors que des souches bactériennes plus éloignées peuvent être, d'un point de vue observationnel, identiques. Ces souches bactériennes étant particulièrement virulentes sur certaines cultures, la mise au point d'un test de diagnostic fiable et simple constitue un enjeu important, y compris du point de vue économique. La problématique consiste alors à construire un modèle d'identification d'un ensemble de groupes de souches bactérienne en utilisant un des critères (caractères génomiques ou autres) permettant de déterminer l'appartenance d'une souche à l'un de ces groupes. Une collection de bactéries phytopathogènes est en effet disponible et permet donc de disposer de jeux de données de référence qui permettent d'envisager la construction de tels modèles d'identification.

Notons que les approches classiques de classification, dont la plupart sont statistiques, ne répondent que partiellement à ce problème d'identification. En effet, comme nous le verrons par la suite, la problématique posée est de fournir une caractérisation exacte qui minimise le nombre de caractères utilisés (pour des raisons de coût mais aussi de fiabilité des procédures de test). Ainsi, nous avons été amenés, en étroite collaboration avec nos collègues de biologie, à définir le problème de caractérisation multiple, afin de pouvoir proposer à la fois un modèle et des outils de calcul efficaces.

Dans la section suivante, nous décrivons ainsi plus précisément les difficultés liées à l'identification précise des souches bactériennes que nous venons d'évoquer.

1.2 Identification de pathovars

Certaines souches bactériennes du genre *Xanthomonas* sont responsables de pathologies présentes sur les cultures principalement maraîchères [Mhedbi, 2010]. Dans ce contexte, un pathovar (en abrégé *pv.*) est défini comme un groupe de souches bactériennes pouvant appartenir à des espèces différentes mais qui provoquent, pour un hôte particulier (ici des végétaux), la même pathologie observée. Les dégâts occasionnés ont un impact non négligeable sur l'économie et, depuis de nombreuses années, des réglementations sont mises en place pour endiguer et contrôler ces menaces phytosanitaires (y compris au niveau européen). Quelques semences contaminées par ces souches pathogènes suffisent pour que cet organisme bactérien se répande au sein de tout un lot de semences et se développe lors de la croissance de la plante. Il peut en résulter la destruction des récoltes si aucun traitement n'est effectué. La figure 1.1 illustre quelques maladies provoquées par différents pathovars du genre *Xanthomonas*.

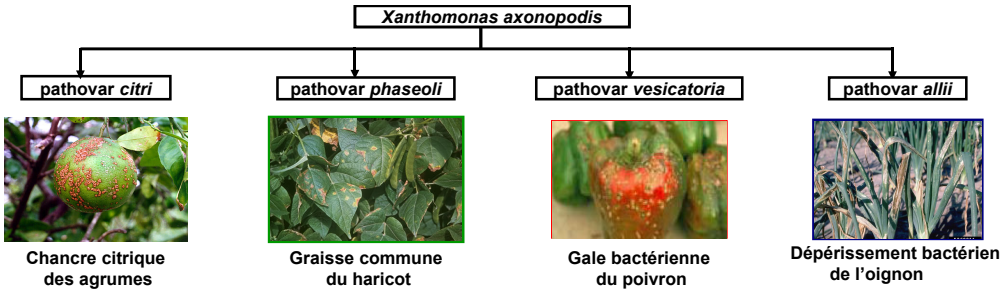


Figure 1.1 : Maladies provoquées par différents pathovars du genre xanthomonas.

Afin de prévenir de tels risques, en amont de la mise en culture des plantations, il existe trois grandes classes de tests de routine qui peuvent être effectués. Au préalable il s'agit d'isoler une colonie de bactéries provenant de la surface des graines, appelée isolat, et d'effectuer un ensemble de sous-tests colorimétriques. Trois techniques sont actuellement utilisées :

- Les tests biochimiques, comme illustrés pour la galerie API 20E® de la société Biomérieux par la figure 1.2[1] , permettent la détection des souches incriminées par l'étude de leurs propriétés biochimiques. Les isolats sont mis en culture sur des milieux différents où l'observation de la dégradation ou de la production d'un composé chimique indique

[1]Source Wikimedia

la manifestation d'un mécanisme métabolique. Le test est donc décomposé en plusieurs sous tests dans lesquels chaque observation de changement de couleur permet de conclure sur la classification de la souche testée.

Figure 1.2 : Exemple de résultat pour l'API 20E® de la société Biomérieux destiné à l'analyse d'enterobactéries.

- Les tests sérologiques ELISA [Engvall and Perlmann, 1972] (voir figure 1.3 [1]) se basent sur la réaction d'anticorps sur les bactéries. Les anticorps sont des protéines produites par les organismes lorsque ceux-ci sont attaqués (par exemple, par des bactéries en se fixant sur leurs antigènes). La mise en évidence d'une souche se fait par la production d'anticorps spécifiques couplés à un substrat fluorescent. La concentration du substrat se mesure donc par coloration. Une forte présence d'antigènes correspond à un échantillon test fortement coloré.

- L'inoculation des bactéries sur la plante cible constitue un dernier moyen de détection de ces bactéries phytopathogènes. Il s'agit d'introduire la souche suspectée sur la plante cible et d'observer l'apparition de signe d'infection.

La taxonomie du genre *Xanthomonas* est sujette à débat et la phylogénie [Schaad et al., 2005, Vauterin et al., 1995], c'est-à-dire l'étude de la proximité génétique des organismes, n'explique pas la spécificité d'hôtes. En effet, une hypothèse serait que les souches génétiquement proches provoquent la même pathologie sur une espèce végétale donnée. Or, il existe des exceptions qui nous amènent à repenser le classement de ces souches par rapport aux maladies qu'elles provoquent. C'est cette classification des bactéries en considérant l'aspect phénotypique de leur pouvoir pathogène qui permet de justifier pleinement le concept de pathovar que nous avons introduit plus haut.

La figure 1.4 illustre la phylogénie de l'espèce *Xanthomonas* classiquement sous forme d'un arbre. Il est intéressant de remarquer que les souches correspondant au pathovar *pv. alfalfae* sont éloignées du point de vue de

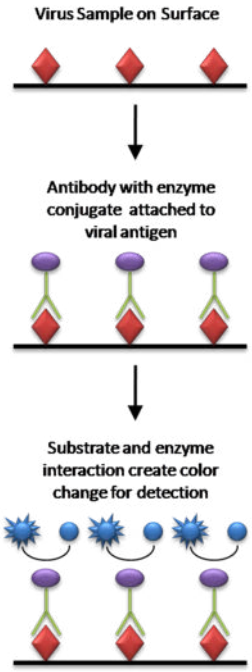


Figure 1.3 : Diagramme explicatif du test ELISA.

cette analyse reposant sur la proximité génétique (une souche est présente dans la partie supérieure de l'arbre alors qu'une autre apparaît dans la partie centrale).

Maintenant que nous venons de voir que s'intéresser de manière générale au profil génétique des souches ne suffit pas, il faut donc extraire plus précisément des caractéristiques génétiques ou phénotypiques plus pertinentes. En effet, dans la construction d'arbres phylogénétiques habituelle, deux souches éloignées peuvent pourtant partager certains gènes communs, qui peuvent s'avérer être les gènes induisant leur comportement pathogène. Il convient donc de disposer de techniques adéquates pour identifier de tels gènes qui pourront ensuite être utilisés pour construire, en les combinant par exemple, à des procédés d'identification plus complexes. Ainsi la question n'est pas de disposer de tout le profil génétique d'un individu mais plutôt de sélectionner un sous-ensemble de caractères discriminants.

1.3 Analyse des souches par PCR

Malgré de forts progrès depuis de nombreuses années pour le séquençage génomique, il reste difficile d'établir une relation entre le génotype, c'est-

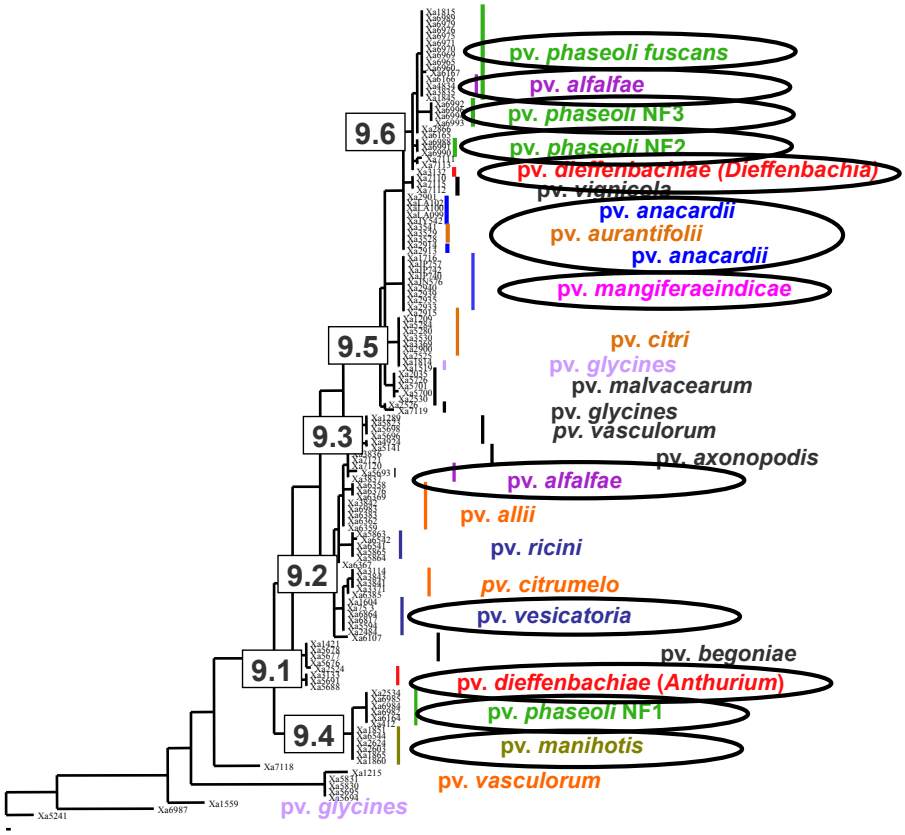


Figure 1.4 : Phylogénie de l'espèce *Xanthomonas*. Les souches du *pv. alfalfae* sont éloignées génétiquement.

à-dire l'information contenue dans l'ADN, et le phénotype (ici la manifestation d'un pouvoir pathogène d'une bactérie). L'identification de gènes repose sur l'expression de séquences particulières qui codent ces gènes. Ainsi, le génome de chaque bactérie est analysé (méthode de séquençage) puis un long processus de classement intervient par l'identification des pathologies inoculées par celles-ci. La technique d'identification décrite dans [Boureau et al., 2013, Hajri et al., 2009] consiste à utiliser une PCR multiplex (réaction en chaîne par polymérase - *polymerase chain reaction*) indiquant la présence ou l'absence de gènes de virulence, qui est inspiré par les tests biochimiques et sérologiques. L'hypothèse est que la présence de certains gènes est corrélée avec une fonction pathogène spécifique. En effet, les études génétiques préliminaires montrent que leur présence joue un rôle dans la différentiation du genre *Xanthomonas*. La PCR est une méthode visant à multiplier l'ADN contenu dans un échantillon afin de faciliter la mise en évidence des gènes. La technique de la PCR se base sur l'ajout d'en-

zyme polymérase dont le rôle est de dupliquer un brin d'ADN. Pour cela l'ADN est extrait des bactéries par destruction de leur membrane puis la solution est mélangée avec des amorces qui s'associent (s'hybrident) au brin préalablement séparé par une augmentation de la température du milieu réactionnel. Après plusieurs cycles de changement de températures, est obtenu un échantillon dont l'ADN a été multiplié. Au bout de plusieurs cycles, des marqueurs sont ajoutés et en fonction d'un seuil pour une concentration, nous considérons l'expression ou non du gène. Par abus de langage, nous considérons qu'un gène est présent si celui-ci est exprimé et absent dans le cas échéant. Ce sont les contraintes imposées par la fabrication d'une puce à ADN (figure 1.5) en terme du nombre limité de gènes à tester et le souci de clarté des résultats, qui nous ont amené à reconsidérer les séquences exhaustives et ont motivé les travaux de cette thèse.

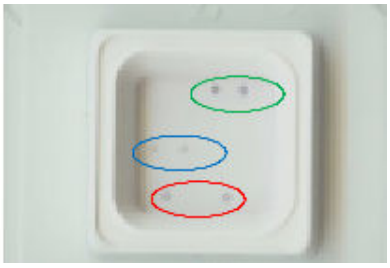


Figure 1.5 : Exemple de résultat sur la macro puce développé par la société Diag Gène. Les cercles indiquent les deux spots redondants associés à l'expression d'un même gène

Ainsi, l'objectif est bien de minimiser le nombre de spots observables qui devront être mis en place sur la puce. Les approches que nous allons ainsi développer, concernant les aspects informatiques de la résolution de ce problème de caractérisation, permettront ainsi de générer une formule de diagnostic (littéralement une formule propositionnelle) qui sera ensuite utilisée en fin de la chaîne de diagnostic.

Nous venons de décrire les enjeux biologiques du problème que nous allons traiter dans cette thèse ainsi que les mécanismes biologiques qui sont mis en œuvre. Du point de vue pratique, il est également nécessaire, dans ce type d'approche, d'avoir toujours à l'esprit, les contraintes mécaniques et physique des méthodes d'acquisition de données pour que les approches étudiées soient à la fois réalistes et pertinentes. C'est ainsi dans ce contexte très pragmatique que le travail décrit dans ce manuscrit a été mené.

1.4 Conclusion

Il est souvent complexe d'établir des collaborations scientifiques à l'intersection de plusieurs disciplines et même comme nous le verrons, à l'intersection de plusieurs domaines au sein d'une même discipline. Le problème de caractérisation multiple, tel que nous l'avons défini, constitue ainsi un exemple possible de mise en commun de savoirs et compétences pour arriver à un résultat concret. Dans ce chapitre, nous venons de voir que la problématique sous-jacente d'identification de souche bactérienne qui nous intéresse est importante à la fois en terme de compréhension du comportement de ces bactéries mais également en terme d'enjeu sanitaire. Les résultats que nous avons ainsi obtenus, et que nous détaillerons dans la dernière partie de cette thèse, sont intéressants du point de vue biologique. Toutefois, au delà de cette application, nous allons voir que la problématique de caractérisation multiple constitue également en elle-même un nouveau défi en matière de modèle de calcul, de part sa complexité importante. Ainsi, l'apport pour la discipline informatique sera, comme nous l'espérons, également enrichissant.

2

Logique propositionnelle, optimisation combinatoire et complexité

Sommaire

2.1 Logique propositionnelle 29
2.1.1 Syntaxe 29
2.1.2 Sémantique 31
2.1.3 Formes normales 33
2.1.4 Fonctions booléennes 36
2.2 Optimisation combinatoire 39
2.2.1 Problèmes d'optimisation combinatoire 39
2.2.2 Panorama des méthodes de résolution 41
2.2.3 Discussions 49
2.3 Quelques notions de complexité 50
2.3.1 Classes de complexité 51
2.3.2 Réductions 54
2.3.3 Autres classes de complexité 56
2.4 Conclusion 60

L'objectif de ce chapitre est de préciser les concepts formels et les modèles que nous utiliserons dans la suite du manuscrit. Après avoir décrit le problème de caractérisation multiple sous son aspect biologique, nous devons

en effet en proposer une modélisation formelle qui permettra de mettre en lumière sa difficulté et ses liens éventuels avec d'autres problèmes existants. Pour cela, nous aborderons la logique propositionnelle qui nous servira de support syntaxique pour l'expression du problème ainsi que des éléments de complexité algorithmique. Dans la mesure où nous allons nous intéresser par la suite à la résolution effective du problème, nous introduirons également les bases des méthodes d'optimisation combinatoire sur lesquelles nous nous appuierons pour proposer de nouveaux algorithmes spécifiques de résolution. Enfin, nous rappellerons également quelques outils statistiques, utilisés dans le traitement des données biologiques, afin de pouvoir clairement positionner notre approche dans l'état de l'art.

2.1 Logique propositionnelle

La logique propositionnelle est la partie de la logique qui traite des formules propositionnelles, qui représentent des assertions qui peuvent être vraies ou fausses. Ces formules sont construites à l'aide de constantes, de variables booléennes et de connecteurs. La logique propositionnelle a été initialement explorée sous ses aspects algébriques et calculatoires par George Boole au 19ème siècle. Le langage ainsi défini permet notamment de représenter simplement des connaissances et c'est ce dont nous allons tirer parti dans cette thèse. Ainsi, nous rappelons ici les principales définitions et concepts que nous utiliserons ensuite lors de la formalisation du problème de caractérisation multiple.

2.1.1 Syntaxe

La logique propositionnelle est syntaxiquement assez simple car elle n'utilise que des constantes, des variables et cinq connecteurs. Cette simplicité syntaxique ne limite toutefois en rien son pouvoir d'expression. Quelques définitions permettent de présenter de manière plus formelle les éléments de la logique propositionnelle.

Le premier élément essentiel de la logique propositionnelle est tout naturellement la notion même de variable propositionnelle (ou booléenne).

Définition 2.1 (Variable booléenne)
Une variable booléenne est une variable qui prend des valeurs de vérité. Nous notons l'ensemble des valeurs de vérité $\mathbb{B} = \{(F)aux, (V)rai\}$ ou $\mathbb{B} = \{0, 1\}$. L'ensemble des variables est noté $\mathcal{X}$.

Afin de construire des formules, il faut également disposer de connecteurs permettant de lier ces variables.

Définition 2.2 (Connecteurs)
Un connecteur est un opérateur logique qui peut être unaire (la négation) ou binaire (la conjonction, la disjonction, l'implication et l'équivalence). Chacun de ces opérateurs est représenté par un symbole particulier :

- négation : $\neg$
- conjonction : $\wedge$
- disjonction : $\vee$
- implication : $\rightarrow$

- équivalence : $\leftrightarrow$

Les opérateurs auxiliaires de parenthésage « (» et «) » peuvent être utilisés afin de lever certaines ambiguïtés lors de l'interprétation.

Définition 2.3 (Littéral)
Un littéral est une variable $x \in \mathcal{X}$ (on parle de littéral positif) ou sa négation $\neg x$ (on parle de littéral négatif). L'ensemble des littéraux est noté $\mathcal{L}$. Une formule est dite atomique si elle n'est constituée que d'un seul littéral.

La combinaison des variables et des connecteurs, auxquels il faut ajouter les deux constantes représentant les valeurs de vérité vrai et faux, permet de définir de manière inductive l'ensemble des formules propositionnelles.

Définition 2.4 (Formule propositionnelle)
L'ensemble des formules propositionnelles $\mathcal{F}$ est défini inductivement comme suit :

- les constantes $\perp$ *et* $\top$ sont des éléments de $\mathcal{F}$,
- tout élément de $\mathcal{X}$ est un élément de $\mathcal{F}$,
- si A est un élément de $\mathcal{F}$ alors $\neg A$ est un élément de $\mathcal{F}$,
- si A est un élément de $\mathcal{F}$ alors (A) est un élément de $\mathcal{F}$,
- si A et B sont des éléments de $\mathcal{F}$ alors $A \wedge B$, $A \vee B$, $A \rightarrow B$ et $A \leftrightarrow B$ sont des éléments de $\mathcal{F}$.

Pour l'étude du problème de caractérisation multiple, les notions de longueur et taille de formules sont nécessaires. Ces notions étant souvent confondues dans la littérature nous donnons donc les deux définitions.

Définition 2.5 (Longueur d'une formule)
La longueur d'une formule, notée $l(\phi)$, est le nombre d'occurrences de variables qu'elle contient.

Définition 2.6 (Taille d'une formule)
La taille d'une formule, notée $var(\phi)$, est le nombre de variables distinctes qu'elle contient.

Dans la suite, nous utiliserons la dernière définition avec la définition $|\phi| = |var(\phi)|$.

2.1.2 Sémantique

Après avoir présenté la syntaxe de la logique propositionnelle, nous devons lui associer une sémantique. Cette sémantique est définie via la notion d'interprétation ou d'affectation des variables propositionnelles qui peut être étendue à une formule quelconque. Il faut donc de manière totale ou partielle déterminer quelle sera la valeur de vérité des variables de la formule. Notons que nous choisissons ici de préférer le terme affectation pour définir l'interprétation des variables propositionnelles.

Définition 2.7 (Affectation)
Une affectation pour une formule ϕ est une fonction $A : \mathcal{X}_\phi \mapsto \mathbb{B}$ où $\mathcal{X}_\phi \subseteq \mathcal{X}$ est l'ensemble des variables de ϕ.

Si toutes les variables ne sont pas instanciées, il s'agit d'une affectation partielle sinon c'est une affectation complète. L'ensemble des affectations possibles pour un ensemble de variables $\mathcal{X}$ est noté $\mathcal{A}$ et a pour cardinal $2^{card(\mathcal{X})}$. Une variable instanciée est par conséquent une variable ayant une valeur de vérité.

Nous pouvons naturellement étendre de manière canonique une affectation des variables en une interprétation des formules construites avec ces variables. Pour cela il suffit de définir la manière dont les connecteurs et les constantes seront interprétés.

Définition 2.8 (Interprétation)
Étant donné une affectation A, l'interprétation qui lui est associée est une fonction $I_A : \mathcal{F} \mapsto \mathbb{B}$ avec les caractéristiques suivantes :

- Pour les symboles constants nous avons $I_A(\top) = V$ et $I_A(\bot) = F$
- Pour toute variable $x \in \mathcal{X}_\phi, I_A(x) = A(x)$
- Les connecteurs logiques sont interprétés comme le montre la table de vérité suivante (tableau 2.1)

ϕ	ϕ'	$\neg\phi$	$\phi \wedge \phi'$	$\phi \vee \phi'$	$\phi \rightarrow \phi'$	$\phi \leftrightarrow \phi'$
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	F	V	F	F	V	V

Tableau 2.1 : Interprétation des connecteurs

Nous noterons $\mathcal{I}$ l'ensemble des interprétations.

Remarque 2.1
À partir de maintenant il est clair que l'affectation $A \in \mathcal{A}$ et son interprétation implicite $I_A \in \mathcal{I}$ peuvent être manipulées de manière équivalente.

Nous pouvons définir la notion de satisfaction et de modèle d'une formule propositionnelle.

Définition 2.9 (Satisfait, falsifie, modèle)

- Une interprétation I satisfait (resp. falsifie) ϕ ssi $I(\phi) = V$ (resp. $I(\phi) = F$).
- Si I satisfait ϕ, alors I est un modèle de ϕ noté $I \models \phi$.
- Si I falsifie ϕ, alors I est un contre-modèle de ϕ noté $I \not\models \phi$.

Exemple 2.1
Si $A = [x_1 = V, x_2 = V, x_3 = F]$ et $\phi = a \wedge b \wedge \neg c$ alors $A \models \phi$ car $I(\phi) = I(V \wedge V \wedge \neg F) = I(V \wedge V) = V$

Il est également nécessaire de parler par la suite de formule satisfiable ou insatisfiable relativement aux affectations possibles des variables booléennes qui la composent. En effet, d'un point de vue calculatoire, résoudre un problème modélisé sous forme propositionnelle revient concrètement à trouver une affectation de valeurs de vérité aux variables qui satisfait la formule. Rappelons que sur le plan du calcul, la logique propositionnelle est décidable, c'est-à-dire qu'il existe un algorithme permettant de déterminer si n'importe quelle formule est satisfiable ou non.

Définition 2.10 (Formule satisfiable, insatisfiable)
Une formule ϕ est dite

- Satisfiable si ϕ possède au moins un modèle ($\exists A \in \mathcal{A}, I_A \models \phi$).
- Insatisfiable si ϕ ne possède aucun modèle ($\forall A \in \mathcal{A}, I_A \not\models \phi$).

Nous pouvons également nous intéresser aux relations qui peuvent exister entre les formules propositionnelles.

Définition 2.11 (Conséquences logiques, formules équivalentes)

- Une formule ϕ est une conséquence logique de ϕ' (noté $\phi' \models \phi$) si tout modèle de ϕ' est un modèle de ϕ ($\forall A \in \mathcal{S}, A \models \phi'$, nous avons $A \models \phi$).

- Deux formules ϕ et ϕ' sont équivalentes (noté $\phi' \equiv \phi$) ssi $\phi' \models \phi$ et $\phi \models \phi'$.

À partir des définitions nous pouvons déduire une équivalence pour les formules suivantes

Proposition 2.1
Soient ϕ, ψ et χ trois formules propositionnelles.

- $\neg\neg\phi \equiv \phi$ (involution de la négation)
- $\phi \vee \phi \equiv \phi$ et $\phi \wedge \phi \equiv \phi$ (idempotence)
- $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$
- $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$
- $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$ et $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$ (de Morgan)
- $\phi \vee (\psi \wedge \chi) \equiv (\phi \vee \psi) \wedge (\phi \vee \chi)$ et $\phi \wedge (\psi \vee \chi) \equiv (\phi \wedge \psi) \vee (\phi \wedge \chi)$ (distributivité)

2.1.3 Formes normales

Nous nous penchons à présent sur la morphologie syntaxique des formules booléennes. Ceci permet en effet de réduire le langage utilisé pour la modélisation de problème afin d'en simplifier par la suite le traitement opérationnel. En effet, il existe pour la logique propositionnelle des formes normales ou canoniques obtenues par réduction de n'importe quelle formule. Nous allons donc passer maintenant en revue un certain nombre de définitions qui nous permettront en particulier d'introduire la notion bien connue de forme normale pour les formules propositionnelles.

Définition 2.12 (Formule positive)
Une formule est positive si elle ne contient que des littéraux positifs. Nous notons $\mathcal{F}_+$ l'ensemble des formules positives.

Une autre restriction sur le langage $\mathcal{F}$ est d'imposer l'utilisation d'un seul connecteur.

Définition 2.13 (Formule conjonctive - $\mathcal{F}^\wedge$)
Une formule est une formule conjonctive si elle ne contient que des conjonctions. L'ensemble des formules conjonctives est noté $\mathcal{F}^\wedge$ et $\mathcal{F}^\wedge_+$ désigne l'ensemble des formules conjonctives positives.

Définition 2.14 (Formule disjonctive - $\mathcal{F}^{\vee}$)
Une formule est une formule disjonctive si elle ne contient que des disjonctions. L'ensemble des formules disjonctives est noté $\mathcal{F}^{\vee}$ et $\mathcal{F}^{\vee}_{+}$ désigne l'ensemble des formules disjonctives positives.

Définition 2.15 (Clause, clause unitaire, clause positive, clause de HORN)

- Une clause est une disjonction de littéraux.
- Une clause est unitaire si elle ne contient qu'un seul littéral.
- Une clause est dite positive si elle ne contient que des littéraux positifs.
- Une clause est une clause de Horn s'il elle contient au plus un littéral positif.

Définition 2.16 (Formule de Horn)
Une formule propositionnelle est sous forme de Horn si c'est une conjonction de clauses de Horn. Nous notons $\mathcal{F}^{Horn}$ l'ensemble des formules sous forme de Horn.

La restriction syntaxique la plus couramment utilisée en logique propositionnelle est la forme normale conjonctive (qui sous-tend en particulier la structure des principaux solveurs SAT). Seuls les connecteurs $\wedge$ et $\vee$ ainsi que les littéraux composent ces formes normales.

Définition 2.17 (Formule normale conjonctive - CNF)
Une formule propositionnelle est en forme normale conjonctive (notée CNF) si c'est une conjonction de clauses. Nous notons $\mathcal{F}^{CNF}$ l'ensemble des CNF.

Remarquons qu'une formule propositionnelle en CNF est vraie si et seulement si toutes ses clauses sont vraies. Une formule en CNF peut donc être vue comme un ensemble de clauses devant toutes être vraies pour que la formule soit vraie.

La définition suivante complète la subsomption des clauses unitaires. Nous rappelons qu'une clause c subsomme une clause c' si et seulement si $\mathcal{L}'^{c} \subseteq \mathcal{L}^{c}$. Dans ce cas c' peut être éliminé de la formule.

Définition 2.18 (CNF u(nitaire)-redondante)
Une CNF est *u-redondante* si au moins une variable contenue dans une de ses clauses unitaires apparaît au moins deux fois dans la CNF.

Le terme « redondant » indique ici que la formule sous forme CNF est équivalente à une autre CNF de longueur plus petite. Il existe deux cas où une CNF est u-redondante :

Soit x,y et z des variables propositionnelles :

- Même parité : si la CNF est de la forme $x \wedge \ldots \wedge (x \vee w \vee \ldots \vee z)$ alors elle est équivalente à $x \wedge \ldots$
- Parité contraire : si la CNF est de la forme $\neg x \wedge \ldots \wedge (x \vee w \vee \ldots \vee z)$ alors elle est équivalente à $\neg x \wedge \ldots \wedge (w \vee \ldots \vee z)$.

Une alternative, que nous pourrions qualifier de « duale », consiste à considérer des formes normales disjonctives.

Définition 2.19 (Cube, cube unitaire, cube positif)

- Un cube est une conjonction de littéraux.
- Un cube est unitaire s'il ne contient qu'un seul littéral.
- Un cube est dit positif s'il ne contient que des littéraux positifs.

Remarque 2.2
Le mot « terme » peut remplacer celui de « cube ».

Définition 2.20 (Formule normale disjonctive - DNF)
Une formule propositionnelle est en forme normale disjonctive (notée DNF) si c'est une disjonction de cubes. Comme pour les CNF, seuls les connecteurs $\vee$ et $\wedge$ ainsi que les littéraux composent une formule DNF. Nous notons $\mathcal{F}^{DNF}$ l'ensemble des DNF.

Définition 2.21 (Impliquant, impliquant premier)
Un impliquant d'une formule ϕ est un cube ϕ' tel que $\phi' \rightarrow \phi$. Il est dit premier s'il n'existe aucun cube ϕ'' de taille plus petite que ϕ' et tel que $\phi'' \rightarrow \phi$.

Ces deux formulations sont équivalentes dans leur pouvoir d'expression.

Proposition 2.2
Toute formule appartenant à $\mathcal{F}$ peut se transformer en CNF ou en DNF.

Une méthode de transformation est de remplacer tout connecteur $\rightarrow$ et $\leftrightarrow$ pour obtenir une formule n'ayant que les connecteurs $\vee$ et $\wedge$ et $\neg$. Puis nous appliquons les lois de De Morgan pour migrer $\neg$ sur les variables. Enfin, nous terminons cette transformation en appliquant les lois de distribution.

Dans cette thèse, nous ne nous intéressons pas à trouver des modèles d'une formule - d'où la distance que nous prenons avec le problème SAT

- mais plutôt à construire des formules dont nous connaissons un modèle donné. Par conséquent, nous allons être amenés à comparer des formules entre elles (selon leur syntaxe).

Dans ce contexte, les tables de vérité, avant de concerner les formules propositionnelles, sont directement liées à la notion plus générale de fonction booléenne.

2.1.4 Fonctions booléennes

Une fonction booléenne est une fonction définie par $f : \mathbb{B}^n \mapsto \mathbb{B}$, d'arité n. Un moyen simple de décrire une fonction booléenne est donc d'en donner la table de vérité, qui correspond à sa définition en extension. L'étude des fonctions booléennes est abordée dans le domaine de la cryptographie sur un plan algébrique. Les fonctions booléennes sont aussi appelées fonctions logiques dans le cadre de la synthèse de circuit électronique [Bryant, 1992].

Exemple 2.2
Soient x_1, x_2 et x_3 des variables propositionnelles et f une fonction booléenne à 3 variables (figure 2.2).

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
F	F	F	F
F	F	V	V
F	V	F	F
F	V	V	F
V	F	F	F
V	F	V	V
V	V	F	F
V	V	V	V

Diagramme de décisions binaires

Lorsque le nombre de variables augmente, il devient nécessaire de proposer d'autres représentations syntaxiques de ces fonctions booléennes. Le plus souvent, les fonctions booléennes sont représentées sous forme de diagrammes de décisions binaires appelés *BDD - Binary Decision Diagram* [Bryant, 1986, Brace et al., 1990].

En instaurant un ordre sur les variables nous obtenons des *ROBDD-Reduced Ordered Binary Decision Diagram* qui représentent la forme canonique d'une BDD (figure 2.1). Toutefois, trouver l'ordre pour lequel

la ROBDD est minimale est NP-difficile [Meinel and Slobodová, 1994, Bollig and Wegener, 1996].

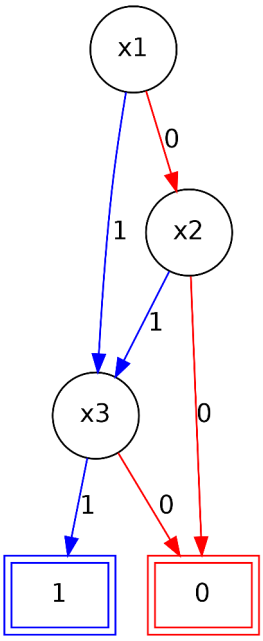


Figure 2.1 : Fonction en ROBDD

Une formule propositionnelle est également un moyen naturel de représenter une fonction booléenne. La notion de représentation est bien évidemment liée à la notion de modèle, c'est-à-dire que la fonction booléenne, vue comme une affectation, est un modèle de la formule.

Nous pouvons immédiatement remarquer que les restrictions syntaxiques sur les formules que nous avons introduites précédemment via les notions de formes normales prennent tout leur sens ici. En effet, étant donnée une fonction booléenne, nous pouvons construire des formules très différentes pour la représenter. Cette problématique constitue d'ailleurs un vaste domaine de recherche qui reste en dehors du champ de cette thèse.

Fonctions booléennes partiellement définies (pdBfs)

Nous allons nous focaliser à présent sur une formalisation générale des formules booléennes qui sera complètement en phase avec la problématique de modélisation de données qui nous intéresse. Nous introduisons donc la notion de fonction booléenne partiellement définie qui sera notée pdBf.

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).

Définition 2.22 (pdBfs)
Une fonction booléenne partiellement définie pf d'arité n 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 $pf(e) = V$ (resp. $pf(e') = F$) et $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) est appelé l'ensemble des exemples positifs (resp. négatifs).

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

Proposition 2.3 (Consistance et inconsistance d'une pdBf)
Une *pdBf* est consistante (resp. inconsistante) ssi $\mathcal{C}^+ \cap \mathcal{C}^- = \emptyset$ (resp. $\mathcal{C}^+ \cap \mathcal{C}^- \neq \emptyset$).

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 est une extension facilement calculable à partir de la classe $\mathcal{C}^+$.

Définition 2.23 (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 présente, on remarque que l'arité d'une *pdBf* est égale à la taille de son extension. Dans [Makino et al., 1999, Eiter et al., 2002], les auteurs développent plusieurs techniques en temps polynomial pour trouver des extensions pour de nombreuses catégories définies de formules. L'extension naturelle vers $\mathcal{F}^{DNF}$ se traduit simplement en utilisant les exemples positifs. Chaque cube est alors construit de telle manière que x_i apparaît si la valuation est positive dans l'exemple positif considéré et dans le cas contraire c'est $\neg x_i$ qui apparaît. Pour une extension vers $\mathcal{F}^{CNF}$, les clauses sont construites de façon duale aux cubes d'une DNF.

Exemple 2.3
Soient x_1, x_2, x_3 des variables propositionnelles et pf une pdBf à 3 variables (figure 2.3).

Une extension en DNF de pf est :

$$(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

x_1	x_2	x_3	$pf(x_1, x_2, x_3)$
F	F	F	$\mathcal{C}^-$
F	V	F	
F	V	V	
V	F	V	$\mathcal{C}^+$
F	F	V	
V	V	V	

Figure 2.2 : Les affectations de (V,F,F) et (V,V,F) ne sont pas définies pour pf.

Disposant d'un formalisme logique pour la modélisation des problèmes que nous allons aborder dans cette thèse, nous nous intéressons à présent aux aspects calculatoires liés à ces modèles.

2.2 Optimisation combinatoire

L'optimisation combinatoire est un domaine de recherche que l'on positionne habituellement à la croisée des mathématiques appliquées, de la recherche opérationnelle et de l'intelligence artificielle, en fonction des modèles et des approches de résolution mis en œuvre. À la différence des problèmes d'optimisation globale qui désignent classiquement les problèmes dont les variables sont continues, les problèmes d'optimisation combinatoires [Papadimitriou and Steiglitz, 1982] considèrent des variables discrètes dont le domaine de valeurs est fini. On trouve ainsi un nombre d'applications important dans le domaine de la logistique ou de la planification. En terme de modélisation, certains problèmes peuvent partager des caractéristiques communes, par exemple leurs solutions admissibles peuvent être des permutations d'un ensemble $\{1, \ldots, n\}$. De nombreuses bibliothèques recensent ces problèmes combinatoires [Beasley, 1990, Reinelt, 1991, Gent and Walsh, 1999].

2.2.1 Problèmes d'optimisation combinatoire

Un problème combinatoire est défini par un ensemble de variables pouvant prendre leurs valeurs dans des domaines. Le produit cartésien de ces domaines définit l'ensemble de toutes les configurations possibles du problème. Un ensemble de contraintes, qui sont des relations sur ces variables, définit l'ensemble des solutions dites admissibles. On peut alors s'intéresser au problème de décision consistant à déterminer s'il existe une solution ad-

missible. Notons que la résolution de ce problème ne nécessite pas obligatoirement le calcul explicite de solutions admissibles. Il est également possible d'adjoindre au modèle précédent une fonction dite objectif permettant de quantifier la qualité des solutions admissibles. Le problème d'optimisation alors induit consiste à déterminer une solution de qualité optimale. Notons que des extensions de ces deux questions fondamentales de l'optimisation combinatoire peuvent être envisagées : calculer toutes les solutions admissibles, prouver qu'il n'existe pas de solution admissible, calculer toutes les solutions optimales...

Nous pouvons formaliser ces notions de manière simple. Pour conserver un discours général, nous ne considérerons pas explicitement les variables, domaines et contraintes d'un problème. De plus, nous considérons ici des problèmes d'optimisation qui peuvent être considérés, par exemple, comme des problèmes où l'on cherche à minimiser le nombre des contraintes violées par une configuration possible.

Définition 2.24 (Problème d'optimisation)
Une instance problème d'optimisation est définie par un ensemble de solutions admissibles $\mathcal{S}$, appelé espace de recherche. La qualité (ou coût) d'une solution est définie par une fonction objectif $f : \mathcal{S} \mapsto \mathbb{R}$.

La notion de solution optimale découle alors naturellement de ce qui précède.

Définition 2.25 (Solution optimale)
Une solution $s* \in \mathcal{S}$ est dite minimale (resp. maximale) si et seulement si $\forall s \in \mathcal{S}, f(s^*) \leq f(s)$ (resp. $f(s^*) \geq f(s)$).

La fonction objectif doit être soit maximisée (problème de maximisation), soit minimisée (problème de minimisation). Tout problème peut être converti dans l'un ou l'autre point de vue.

D'un point de vue calculatoire, malheureusement, les problèmes de décision associés à ces problèmes combinatoires sont connus pour être difficiles à résoudre. Il en va naturellement de même pour les problèmes d'optimisation. Depuis de nombreuses années, plusieurs résultats indiquent que ces problèmes sont généralement NP-difficil [Garey and Johnson, 1979]. Il est alors évident que des méthodes d'exploration exhaustive de l'espace des solutions, que ce soit pour trouver une solution admissible ou une solution optimale, peuvent dans le pire cas (sous l'hypothèse de complexité $\mathbf{P} \neq \mathbf{NP}$) entraîner des calculs dont le temps sera exponentiel. Dès lors, de nombreuses approches ont été proposées pour aborder ces problèmes de manière efficace.

Un focus sur le problème SAT

Afin d'illustrer notre propos sur un exemple plus concret, en relation avec la première partie de ce chapitre, nous présentons dans un premier temps le problème de satisfaction en logique propositionnelle (SAT). En effet, même si, comme nous l'avons mentionné, il ne constitue pas réellement le cœur de notre travail, il permet de mettre en avant les principaux aspects des difficultés qui ont été historiquement rencontrées lors de la résolution de problèmes combinatoires.

Rappelons que le problème SAT est un problème de décision. Pour une formule ϕ de $\mathcal{F}^{CNF}$, il s'agit de déterminer si ϕ est satisfiable. Historiquement Stephen Cook ([Cook, 1971]) démontra la NP-complétude du sous-problème 3SAT, où toutes les clauses contiennent exactement 3 littéraux. Comme il est possible de transformer polynomialement n'importe quelle formule ϕ au format 3SAT, le résultat précédemment est donc généralisé à SAT. Cependant, SAT devient P-Complet si la restriction de ϕ est 2SAT ou bien encore HORNSAT. Ceci montre donc qu'il faut soigneusement étudier le pouvoir d'expression du langage de modélisation (ici une description clausale d'un problème) pour pouvoir inférer la difficulté réelle du calcul qui sera induit.

Le problème SAT peut se décliner en plusieurs variantes :

- Une des variantes de SAT les plus connues est sans nul doute MAX-SAT qui consiste à trouver une affectation qui satisfait le plus grand nombre de clauses de la formule [Hansen and Jaumard, 1990].

- Une autre variante MINONES(SAT)[Kratsch and Wahlström, 2010] consiste à trouver une affectation valide minimisant le nombre de variables à vrai.

L'étude des problèmes d'optimisation combinatoire a conduit au développement de diverses approches de résolution dont nous dressons un panorama dans la section suivante.

2.2.2 Panorama des méthodes de résolution

Dans la littérature, on distingue habituellement deux grandes classes de méthodes pour la résolution des problèmes combinatoires (figure 2.3). Selon que l'on s'intéresse à un problème de satisfaction ou d'optimisation, leur principales caractéristiques peuvent être résumées ainsi :

- Problème de décision : (C) capacité à démontrer la satisfiabilité ou l'insatiabilité d'une instance vs. (I) capacité à montrer uniquement la satisfiabilité si c'est le cas.

- Problème d'optimisation : (C) capacité à trouver une solution optimale vs. (I) capacité à trouver une solution de bonne qualité.

Notons que ces propriétés sont obtenues au prix de coûts calculatoires différents, en raison de la complexité algorithmique déjà évoquée. Ainsi, la classification usuelle propose de distinguer les méthodes complètes (ayant les caractéristiques notées (C)) plus haut et les méthodes incomplètes (ayant les caractéristiques notées (I)).

Les méthodes complètes (aussi appelées exactes) garantissent la propriété d'optimalité de la solution trouvée et sont généralement des méthodes exhaustives dans le cas où toutes les solutions sont implicitement explorées. Par exhaustive, nous entendons que la méthode peut visiter dans le pire des cas l'intégralité de l'espace de recherche ce qui peut aboutir à une complexité exponentielle en terme de temps. Cependant des améliorations tentent d'écarter les zones infructueuses pendant la recherche en instaurant par exemple un mécanisme d'élagage comme dans les algorithmes de type *branch and bound*.

Les méthodes incomplètes (ou approchées) désignent l'ensemble des méthodes dont d'optimalité n'est pas garantie. L'idée est d'instaurer un compromis entre la qualité (coût) de la solution optimale et le temps de résolution que nous souhaitons le plus court possible. Ainsi, depuis les années 1970, nous assistons à l'émergence des heuristiques pour la résolution de problèmes NP-Difficiles. Selon [Papadimitriou and Steiglitz, 1982] une heuristique est vue comme une méthode de résolution spécifique à un problème sans garantie formelle sur la qualité de la solution mais qui en pratique révèle sa robustesse. Par extension, les métaheuristiques [Glover, 1986] sont des méthodes génériques indépendantes du problème à traiter.

Historiquement, au sein des métaheuristiques nous distinguons les algorithmes de recherche locale dont le but est de réparer une solution d'étape en étape par un voisinage et les algorithmes à base de population, particulièrement les algorithmes évolutionnaires manipulant un ensemble de solutions qui sont combinées et mutées à chaque itération.

Dans les sous sections suivantes nous nous concentrons sur les méthodes qui ont été utilisées dans cette thèse. Nous présentons ainsi le principe du *branch and bound* puis des approches métaheuristiques.

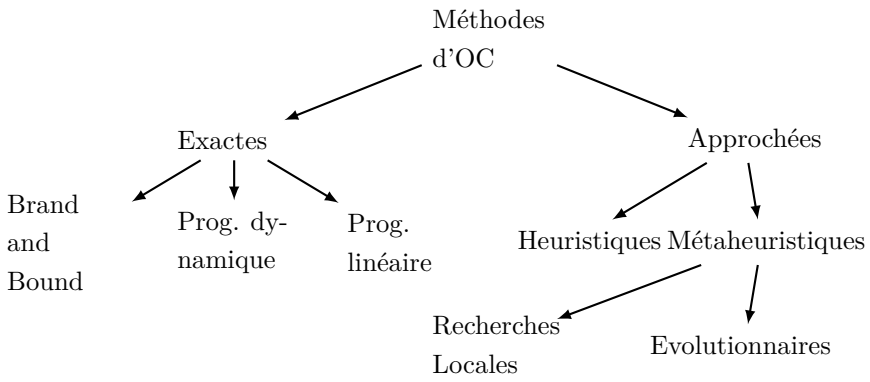


Figure 2.3 : Panorama des techniques d'optimisation combinatoire

Branch and bound

Un algorithme de *branch and bound* (branchement et évaluation) est basé sur le principe du *backtracking* (retour arrière)[Land and Doig, 1960] (figure 2.4). Le backtracking a pour but de construire une solution valide incrémentalement en remettant en cause les choix effectués précédemment. Il permet d'explorer l'espace de recherche sous forme d'arborescence en essayant d'écarter rapidement les solutions partielles non candidates. L'exploration de cette arborescence est faite soit en largeur d'abord où tous les nœuds d'un niveau sont visités avant de passer au niveau suivant, soit en profondeur d'abord qui consiste à descendre au plus bas de l'arbre de l'exploration avant de reconsidérer nos choix. Il existe des heuristiques plus sophistiqués en se basant sur des stratégies comme visiter le meilleur (nœud) d'abord (*first best*).

En se basant sur l'exploration en profondeur d'abord, le branch and bound élague au plus vite les branches qui n'apporteront aucune meilleure solution. Lors du choix d'une nouvelle variable (branch), une nouvelle solution partielle est créée à partir de laquelle les bornes (bound) inférieures et supérieures sont mises à jour. Le mécanisme d'élagage intervient dans les branchements suivants lorsque ces bornes se rejoignent indiquant l'impossibilité d'une amélioration dans le sous arbre courant. Le branch and bound se termine lorsque n'y a plus de possibilité de branchement.

Recherche locale

Les méthodes de recherche locale sont des métaheuristiques basées sur l'application itérative d'opérateurs à partir d'une solution initiale afin d'obtenir la solution optimale. Un opérateur est une fonction qui modifie la solution

Figure 2.4 : Exemple d'exploration avec backtraking sur le problème des n-queens ($n = 4$)

courante en une nouvelle solution [Hoos and Stützle, 2004]. Il est défini par une relation de voisinage qui est une relation binaire irréflexive $\mathcal{N} \subseteq 2^{\mathcal{S}}$ et qui est généralement symétrique. L'ensemble des solutions possibles atteignables par l'application d'un opérateur définit alors le voisinage de celui-ci.

Définition 2.26 (Voisinage)
Pour une solution $s \in \mathcal{S}$, son voisinage noté $\mathcal{V} : \mathcal{S} \mapsto 2^{\mathcal{S}}$ est défini par $\mathcal{V} = \{s' | s' \in \mathcal{S}, \mathcal{N}(s, s')\}$. Le passage d'une solution s à une solution voisine est appelé *mouvement*.

En s'appuyant sur la notion de voisinage, il est alors possible de définir la notion d'optimum local. Les approches incomplètes reposent souvent sur des heuristiques permettant d'atteindre rapidement un optimum local (c'est-à-dire, une solution de meilleure qualité par rapport à ses voisins) en pratiquant des améliorations successives. Afin d'échapper aux optimums locaux, des heuristiques permettent d'explorer d'autres zones de l'espace de recherche en vue d'y trouver un meilleur optimum local. L'objectif est alors d'alterner habilement ces heuristiques pour atteindre, dans l'idéal, un optimum global.

Définition 2.27 (Optimum local, plateau)
Un optimum local est une solution s^* telle que pour toute solution voisine $s' \in \mathcal{V}(s^*)$ nous constatons que $f(s^*) \leq f(s')$. Nous utilisons le terme de plateau si nous avons l'égalité stricte.

Une stratégie d'exploration efficace de l'espace de recherche repose donc sur un compromis entre les mécanismes de diversification et d'exploitation. En effet, il devient alors nécessaire d'établir d'une part un parcours privilégiant la qualité de la solution au risque de tomber dans un optimum local, et d'autre part de garantir un échantillonnage suffisant de l'espace de recherche.

L'architecture de base d'un algorithme de recherche locale est structuré par :

- Une phase d'initialisation : Il s'agit de générer une solution aléatoire ou de construire par une heuristique une solution initiale. Dans la suite nous supposons que la solution initiale est une donnée en entrée.

- Une fonction d'évaluation : Cette fonction mesure la qualité des solutions (aussi appelée fonction de coût). En pratique nous retrouvons souvent comme fonction d'évaluation, la fonction objectif du problème d'optimisation traité.

- Un opérateur et son voisinage : Un opérateur est bien souvent associé à une heuristique de sélection d'individu qui extrait une seule solution dans le voisinage. Nous pouvons privilégier par exemple une solution voisine choisie aléatoirement ou considérer la première solution voisine améliorante.

- Un critère d'arrêt : La procédure de recherche s'arrête après un nombre d'itérations prédéfini ou lorsqu'une solution optimale est atteinte.

Il existe de nombreux algorithmes de recherche locale. Nous nous intéressons plus particulièrement à la recherche Tabu et VNS qui ont influencé nos développements. Avant cela, nous rappelons l'algorithme de recherche locale canonique basé sur l'unique idée d'améliorations successives.

Algorithme de descente

La terminologie descente est ici utilisée en référence à un problème de minimisation dans lequel il s'agit donc de faire décroître progressivement la fonction objectif en visitant des configurations voisines. Notons que la terminologie anglo-saxonne se base elle sur les problèmes de maximisation en utilisant le terme de *Hill climbing*.

Une formulation simple de cet algorithme pour un problème de minimisation est la suivante.

Algorithme 1 : Algorithme de Descente

Données : s une solution initiale, f une fonction objectif et un voisinage $\mathcal{V}$

Résultat : s^* une solution optimale locale

$s^* \leftarrow s$;

répéter

  $s \leftarrow \text{Selection}(\mathcal{V}(s^*))$;

  si $f(s^*) > f(s)$ **alors**

    $s^* \leftarrow s$;

  fin

jusqu'à $f(s^*) < f(s)$;

Évidemment, un tel algorithme s'arrête dès qu'un optimum local est rencontré. En fonction du paysage de l'espace recherche et du point de départ fourni par l'initialisation, il est alors possible ou non d'atteindre un optimum global.

Une stratégie de diversification, qui consiste à l'exploration d'autres zones de l'espace de recherche, peut résider simplement à l'action d'un mécanisme de redémarrage aléatoire. Il suffit alors de repartir d'une nouvelle configuration au bout d'un certain nombre d'itérations ou lorsque la recherche stagne (c'est-à-dire, est bloquée dans un optimum local). Une autre stratégie commune de diversification consiste à effectuer, selon une certaine probabilité, des mouvements aléatoires qui peuvent dégrader la solution courante. En effet, nous voyons que ces deux propositions très simples induisent inévitablement des paramètres (fréquence du redémarrage, taux d'acceptation de mouvements aléatoires) qui régissent le comportement de l'algorithme en contrôlant, plus ou moins finement, la balance entre diversification et intensification de la recherche, véritable clé de voûte de ces métaheuristiques. Évidemment, l'introduction de ces paramètres pose immédiatement la question de leur réglage et de l'adéquation de ce réglage avec l'instance ou la famille d'instances de problèmes que l'on souhaite résoudre. Nous discuterons de ces aspects en fin de section.

La volonté de rendre les algorithmes de recherche locale plus efficaces a amené les chercheurs à proposer des variantes incluant des mécanismes de contrôle plus sophistiqués. Nous présentons ainsi la méthode *Tabu* basée sur le principe de prohibition de certains mouvements en vue d'éviter à l'algorithme de rester piégé dans un optimum local.

Méthode Tabu

L'algorithme *Tabu* (nous utiliserons ici la terminologie anglo-saxonne), introduit par Fred Glover et Manuel Laguna [Glover and Laguna, 1997], ajoute deux mécanismes à l'algorithme de descente précédent. Tout d'abord, afin de s'extirper d'un optimum local, un mécanisme de perturbation est intégré, autorisant la dégradation de la qualité de la solution courante. De plus, pour éviter de visiter les mêmes solutions, une liste dite « tabou » est utilisée. Comme il est impossible et pas forcement souhaitable de stocker l'ensemble de toutes les configurations parcourues, cette liste ne conserve que les derniers mouvements réalisés lors de la recherche. L'algorithme 2 peut être décrit de la manière suivante :

Algorithme 2 : Algorithme Tabu

Données : s une solution initiale, f une fonction objectif, un voisinage $\mathcal{V}$, une liste tabou T
Résultat : s^* une solution optimale locale
$T \leftarrow \emptyset$;
$s^* \leftarrow s$;
répéter
  $s \leftarrow$ Selection($\mathcal{V}(s^*) \setminus T$);
  Update(T,(s,s^*));
  si $f(s^*) > f(s)$ **alors**
    $s^* \leftarrow s$;
  fin
jusqu'à *condition d'arrêt*;

Le mécanisme du voisinage réactif

D'autres moyens permettent de s'extirper d'un optimum local. On peut par exemple envisager de modifier la fonction d'évaluation par des systèmes de pondération évoluant au cours de la recherche, ce qui a pour effet de modifier le paysage de l'espace de recherche. Il est également possible, toujours pour modifier le paysage de l'espace de recherche, de changer dynamiquement de fonction de voisinage au cours de la recherche. En effet, la notion d'optimum local est complètement liée au voisinage et donc, un optimum local pour un voisinage v_1 n'est pas nécessairement un optimum local pour un voisinage v_2. D'autre part, chaque voisinage peut avoir une granularité différente, par rapport à une distance intrinsèque, si elle existe, entre configurations. Il est donc parfois possible d'organiser plusieurs voisinages au sein d'un ordre et

de structurer la stratégie de recherche en utilisant des voisinages de plus en plus grands, permettant donc une exploration accrue.

VNS est l'acronyme de *Variable Neibourghood Search* [Bräysy, 2003] et désigne une recherche locale faisant appelle à un ensemble de voisinages. L'algorithme 3 décrit ce type de méthode.

Algorithme 3 : Algorithme VNS

Données : s une solution initiale, f une fonction objectif et un ensemble de voisinages $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$
Résultat : s^* une solution optimale locale
$s^* \leftarrow s$;
$k' \leftarrow 1$;
tant que $k' \leq k$ **faire**
  $s \leftarrow \text{Selection}(\mathcal{V}_{k'}(s^*))$;
  si $f(s^*) > f(s)$ **alors**
    $s^* \leftarrow s$;
    $k' \leftarrow 1$;
  sinon
    $k' \leftarrow k' + 1$;
  fin
fin

Comme mentionné plus haut, les voisinages visités sont de plus en plus large, ce qui permet de concentrer la recherche sur une zone réduite puis couvrir si besoin la totalité de l'espace de recherche.

Après avoir rapidement présenté quelques options possibles pour définir un algorithme de recherche locale, nous présentons à présent une autre catégorie d'algorithmes métaheuristiques, qui prennent leur inspiration dans la nature (on parle alors d'algorithmes bio-inspirés).

Les algorithmes génétiques

Les algorithmes évolutionnaires sont les représentants canoniques de ces algorithmes inspirés par la nature. Initialement introduits pour résoudre des problèmes sur les variables réelles [Rechenberg, 1965], la variante désormais la plus connue est sans nul doute les algorithmes génétiques proposés par Holland [Holland, 1975] dans le cadre de l'optimisation combinatoire. À la différence d'une recherche locale, un algorithme génétique manipule non pas une solution mais un ensemble de solutions et propose une alternative aux méthodes de recherche locale pour générer plusieurs solutions finales.

En 1872, Darwin introduit le concept de la sélection naturelle après observation de la faune durant un voyage d'études [Darwin, 1872]. Il remarqua en particulier l'évolution de caractères morphologiques (par exemple le bec d'une espèce) et conclut l'importance de l'environnement sur la sélection des individus. La nature parvient à garantir au fur et à mesure des générations l'émergence d'une population constituée par des individus les plus adaptés à leur milieu et dont les avantages évolutifs sont transmis à la descendance par le mécanisme pérenne de la reproduction. La capacité d'adaptation des individus correspond à la fonction d'évaluation.

Un algorithme génétique est composé de plusieurs opérateurs :

- Opérateur d'initialisation : Il s'agit de générer une population de solutions aléatoires ou à l'aide d'une heuristique.
- Opérateur de croisement : À cette étape, il s'agit de sélectionner deux individus (parents) pour obtenir de nouveaux individus (enfants). Le mécanisme de croisement prend en compte les caractéristiques des parents pour construire les enfants. Dans le cas de vecteurs binaires (utilisés pour SAT), il existe de nombreux croisements. Nous pouvons renvoyer le lecteur vers [Srinivas and Patnaik, 1994] pour plus de détails.
- Opérateur de mutation : Il permet de diversifier la population en introduisant des modifications ponctuelles et aléatoires chez les individus créés.
- Opérateur de sélection : C'est dans cette étape que la sélection naturelle intervient et qu'une stratégie permet de choisir les individus de la prochaine génération.

Comme pour les méthodes de recherche locale, un algorithme génétique contient un critère d'arrêt pouvant dépendre du nombre de générations ou de la qualité de la solution optimale courante.

2.2.3 Discussions

Les problèmes d'optimisation combinatoire sont des problèmes difficiles à résoudre ce qui a mené à reconsidérer les approches traditionnelles (complètes) par l'émergence de métaheuristiques. Dès lors, les méthodes de recherche locale et évolutionnaires proposent un cadre de résolution pertinent et intéressant pour la résolution de problèmes combinatoires. Nous pouvons citer les travaux combinant les deux approches comme GASAT

Données : une instance
Résultat : L'ensemble des meilleurs individus P^*
$P \leftarrow$ InitPopulation();
$P \leftarrow P^*$;
tant que *not StopCondition* **faire**
  $(p_1, p_2) \leftarrow SelectParents(P)$;
  $e \leftarrow Crossover(p_1, p_2)$;
  $e \leftarrow Mutation(e)$;
  $P \leftarrow Update(P, e)$;
  $P^* \leftarrow UpdateBest(P, P^*)$;
fin

[Lardeux et al., 2006] qui reposent sur l'association d'un algorithme génétique et d'une recherche locale pour la résolution du problème SAT.

Une principale difficulté réside dans le paramétrage nécessaire (*tuning*) de ces algorithmes pour obtenir des bonnes performances. Même si ce paramétrage peut être effectué par un expert, des approches complémentaires ont été récemment développées. Les méthodes hors ligne (*offline*) [Nannen and Eiben, 2007, Hutter et al., 2007] qui consistent à sélectionner un jeu de paramètres lors d'une première phase d'apprentissage sur un jeu d'instances et les approches en ligne (*online*) [Hamadi et al., 2011] modifiant ces paramètres au cours de la recherches.

Nous avons défini les problèmes d'optimisation ainsi que quelques méthodes de résolution. Un aspect qui est très important en optimisation combinatoire est la complexité dont nous rappelons quelques éléments dans la section suivante.

2.3 Quelques notions de complexité

La théorie de la complexité est intimement liée à l'étude de la calculabilité et de la décidabilité dans un cadre mathématique et logique. Comme nous l'avons évoqué précédemment, la décidabilité s'intéresse à l'étude de la possibilité d'apporter une réponse en un temps fini à un problème de décision.

Ainsi, un problème de décision est un problème dont les instances peuvent être partitionnées en deux groupes : $\mathcal{I}^+$ qui désigne les instances positives du problème auxquelles nous pouvons répondre par « oui » et $\mathcal{I}^-$ qui désigne les instances négatives du problème auxquelles nous pouvons ré-

pondre par « non ». Démontrer qu'un problème est décidable permet donc d'affirmer l'existence d'un algorithme permettant de répondre à cette question. Comme nous l'avons déjà mentionné, la logique propositionnelle est décidable.

Dès lors, il convient de s'intéresser aux caractéristiques opérationnelles des méthodes algorithmiques que nous allons utiliser dans la suite de cette thèse pour en déterminer la complexité. La notion de complexité algorithmique caractérise ainsi la difficulté des problèmes, plus particulièrement concernant les ressources (espace et temps) nécessaires pour résoudre une instance de problème donnée. De manière générale, le temps de calcul est souvent considéré comme la ressource la plus significative pour un algorithme, la mémoire consommée étant en général très liée. Donc, sauf mention contraire, il sera admis que c'est la complexité temporelle qui est désignée lorsque nous parlons de complexité.

Dans la suite, nous présentons les notions générales de la complexité.

2.3.1 Classes de complexité

La machine de Turing [Turing, 1936] est un modèle abstrait de la notion de calcul dont la thèse de Church garantit l'équivalence à toute fonction calculable. Ce modèle est essentiel en informatique théorique et sert donc de base fondamentale aux notions de complexité algorithmique et de calculabilité.

Une machine de Turing T est composée d' :

- Un ruban de taille infini dans les deux sens découpé en cellules contenant un caractère d'un alphabet fini Γ contenant systématiquement un caractère blanc noté $\square$;
- Une tête de lecture-écriture qui est placée sur une cellule et qui a la possibilité de se déplacer sur la droite (D) ou la gauche (G) ;
- Une unité de commande qui à partir d'un ensemble fini d'états $\mathcal{Q}$ et d'une table de transitions δ, modifie l'état courant q de la machine et déplace la tête de lecture en fonction du caractère lu.

Parmi l'ensemble des états $\mathcal{Q}$, nous distinguons l'état initial désigné par q_0 et un sous-ensemble d'états acceptants $\mathcal{Q}_{acc}$. L'initialisation de la machine est effectuée en inscrivant sur le ruban les données d'entrées codées avec l'alphabet $\mathcal{A}$ et le reste des cellules contiennent $\square$. La machine se trouve placée dans l'état q_0 et pointe sur le premier caractère à lire. Ensuite, la machine exécute le « programme » à l'aide de la table de transitions δ qui d'un état et d'un caractère lu procède à l'écriture d'un nouveau caractère,

du changement d'état courant et indique le déplacement de la tête avec $\delta : \Gamma \times \mathcal{Q} \setminus \mathcal{Q}_{acc} \mapsto \Gamma \times \mathcal{Q} \times \{D, G\}$. Le processus s'exécute jusqu'à l'obtention d'un état acceptant ou d'une transition non définie. Une machine de Turing accepte un mot (donnée) si et seulement si un état d'arrêt est atteint.

En informatique, il est plus courant de calculer une valeur que de décider de répondre par « oui » ou par « non ». Il est tout à fait possible d'extraire une solution du ruban lorsque notre machine accepte un mot. La machine de Turing propose donc une réponse au problème de calculabilité mais malheureusement pour la décidabilité, Gödel démontre qu'il existe des problèmes qui n'apportent pas de réponse en un temps fini, c'est-à-dire qu'ils n'atteignent jamais un état acceptant [Gödel, 1931]. Le problème de l'arrêt, proposé initialement par Hilbert, qui consiste à savoir si une exécution sur une machine de Turing se termine, est connu pour être un problème indécidable.

Pour chaque problème décidable se pose alors la question des ressources nécessaires à la résolution sur une machine T bornée par une fonction $f(n)$, où n désigne la taille des données. Quelles ressources en mémoire sont nécessaires c'est a dire l'espace maximum utilisé lors de l'exécution qu'on note $\mathbf{SPACE}(f(n))$? De même quelles ressources en temps qui dénote le nombre maximum de transitions noté $\mathbf{TIME}(f(n))$ sont nécessaires à la résolution du problème ? C'est sur ce dernier point que portera la complexité.

Intuitivement, pour la comparaison de deux fonctions polynômes, pour une grande valeur de n l'importance du plus haut degré devient évidente. En reprenant cette idée pour n'importe quelles fonctions, nous pouvons introduire une mesure asymptotique se basant sur la notation du grand $\mathcal{O}$ de Landau.

Définition 2.28 (Mesure asymptotique de complexité)
Soit f et g deux fonctions définies $\mathbb{N} \mapsto \mathbb{N}$, nous avons $f(n) = \mathcal{O}(g(n))$ s'il existe une constante $c \in \mathbb{N}$ telle que nous avons pour n suffisamment grand $f(n) \leq c \times g(n)$

Dès lors, nous pouvons établir une hiérarchie sur les degrés de difficulté croissante en temps ou en espace pour un algorithme :

- En $\mathcal{O}(1)$ désigne les exécutions constantes ;
- En $\mathcal{O}(log(n))$ désigne les exécutions logarithmes ;
- En $\mathcal{O}(n)$ désigne les exécutions linéaires ;
- En $\mathcal{O}(n^c)$ avec $c \in \mathbb{N}$ désigne les exécutions polynomiales ;

- En $\mathcal{O}(2^n)$ avec $c \in \mathbb{N}$ désigne les exécutions exponentielles ;

La classe **P** est considérée comme la classe contenant les problèmes faciles à résoudre. Au début des années 1970, Stephen Cook et Leonid Levin proposent d'introduire l'idée de la classe **NP** basée sur la reconnaissance d'un mot par une une machine de Turing non-déterministe (**N** dénote le non-déterminisme) [Cook, 1971, Levin, 1973]. Une machine de Turing non-déterministe est une machine de Turing qui admet plusieurs transitions à partir d'un état.

Plus formellement nous définissons ces deux classes de complexité par :

Définition 2.29 (Classe P)
La classe $\mathbf{P} = \bigcup_{k\geq 1} \mathbf{TIME}(n^k)$ est l'ensemble des problèmes de décision pouvant être résolus en temps polynomial par une machine de Turing déterministe.

Définition 2.30 (Classe NP)
La classe $\mathbf{NP} = \bigcup_{k\geq 1} \mathbf{NTIME}(n^k)$ est l'ensemble des problèmes de décision pouvant être résolus en temps polynomial par une machine de Turing non déterministe.

Si nous considérons l'aspect espace, l'intérêt se portera particulièrement sur la classe **PSPACE**.

Définition 2.31 (Classe PSPACE)
La classe $\mathbf{PSPACE} = \bigcup_{k\geq 1} \mathbf{SPACE}(n^k)$ est l'ensemble des problèmes de décision pouvant être résolus en espace polynomial par une machine de Turing.

Nous pouvons aussi définir une classe **NPSPACE** version non déterministe en espace de la classe **PSPACE** pendant de la classe **NP** vis à vis de la classe **P**. Mais par [Savitch, 1970], nous savons que **PSPACE**=**NPSPACE**, ainsi nous garderons que **PSPACE** comme référence.

Proposition 2.4 (Inclusion des classes)
Nous avons :

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$$

Les problèmes de classe **P** sont reconnaissables en temps polynomial sur une machine de Turing non déterministe donc $\mathbf{P} \subseteq \mathbf{NP}$. Actuellement, $\mathbf{NP} \subseteq \mathbf{P}$ reste une question ouverte mais plusieurs indices laissent penser que la conjecture $\mathbf{P} \overset{?}{\neq} \mathbf{NP}$ est vraie. Ainsi, les problèmes de cette classe sont

considérés comme difficiles et une approche complète peut s'avérer parfois très coûteuse avec des algorithmes de complexité non polynomiale en $\mathcal{O}(2^n)$ par exemple.

Remarque 2.3

- Les définitions précédentes considèrent implicitement l'étude générale dans le pire des cas en calculant pour chaque problème une borne maximale. En pratique pour un sous-ensemble d'instances, il se peut que la complexité soit réduite.

- Nous indiquerons sans ambiguïté qu'un problème est $\mathcal{C}$ pour désigner que ce problème appartient à la classe $\mathcal{C}$.

2.3.2 Réductions

Face à un problème, il est utile de connaître sa difficulté. Plus précisément, nous essayons d'identifier l'appartenance d'un problème à une classe donnée et de trouver une relation en se posant la question suivante :« Un problème Π est-il aussi difficile qu'un problème Π' ? ». Une première méthode consiste donc à revenir à la définition de chaque classe et à montrer l'équivalent du problème avec une machine de Turing spécifique dont la construction peut se relever fastidieuse. Une deuxième méthode consiste à réduire le problème désiré vers un autre. On introduit alors la notion de réduction qui est une relation transitive et réflexive entres les problèmes. Intuitivement il s'agit de montrer qu'il existe une transformation pour recoder toute instance en conservant la propriété de satisfiabilité.

Définition 2.32 (Relation de réduction)
Un problème Π se réduit à Π', noté $\Pi \preccurlyeq_R \Pi'$, s'il existe une fonction f telle que pour toute instance x, $x \in \Pi$ ssi $f(x) \in \Pi'$. De plus si $\Pi' \preccurlyeq_R \Pi$, nous notons $\Pi \equiv_R \Pi'$.

Stephen Cook introduit aussi implicitement la réduction de Turing utilisant une machine à Oracle. Une machine de Turing à Oracle T^A est une machine T qui a la possibilité de questionner une autre machine A en comptant comme une transition. Pour une quelconque classe de complexité $\mathbf{C}$, $\mathbf{C}^A$ est la classe des langages décidés dans un temps borné comme dans $\mathbf{C}$, mais avec l'appel à un oracle A

Définition 2.33 (Oracle)
Soit **C** et **C'** deux classes de complexité en temps et L un langage.

$$\mathbf{C}^{\mathbf{C}'} = \bigcup_{L \in \mathbf{C}'} \mathbf{C}^L$$

Pour la théorie de la NP complétude nous utilisons généralement une réduction de Karp [Karp, 1972], dont la fonction f est calculable en temps polynomial et conserve donc l'appartenance à (P) et (NP).

Remarque 2.4
La définition précédente définit implicitement que la réduction est une fonction *many-one*.

L'inclusion d'un problème dans une classe n'est pas une propriété pertinente d'où la nécessité de définir de manière générale deux concepts clés qui sont la C-difficulté et la C-Complétude.

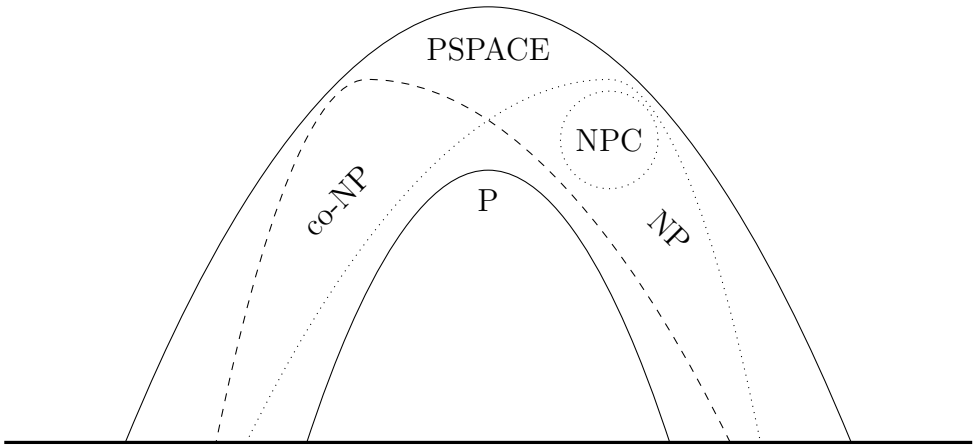
Définition 2.34 (Problème C-difficile)
Soit une classe **C** et une réduction $\preccurlyeq_R$. Un problème Π est **C-difficile** si : $\forall\ \Pi' \in \mathbf{C}$ nous avons $\Pi' \preccurlyeq_R \Pi$

Définition 2.35 (Problème C-Complet)
Soit une classe **C** et une réduction $\preccurlyeq_R$. Un problème Π est **C-Complet** si : $\Pi \in \mathbf{C}$ et $\forall\ \Pi' \in \mathbf{C}$ nous avons $\Pi' \preccurlyeq_R \Pi$

Dans un cas pratique il suffit juste de démontrer qu'il existe une réduction de Π vers un problème Π' car nous garantissons qu'un problème se réduit à tous les autres par la clôture sur la relation de réduction. Enfin, nous pouvons définir les classes complémentaires comme suivant.

Définition 2.36 (Classe complémentaire, problème co-C)
Soit C une classe de problèmes de décision. On définit la classe **co-C** par : $\mathcal{I} = (\mathcal{I}^+, \mathcal{I}^-) \in$**co-C**$\Leftrightarrow \mathcal{I} = (\mathcal{I}^-, \mathcal{I}^+) \in$**C**

La distinction entre la classe **NP** et **co-NP** reste une question ouverte, de même pour $\mathbf{P} \subseteq \mathbf{NP} \cup \mathbf{co-NP}$. Les conjectures actuelles supposent que toutes ces classes sont différentes.



2.3.3 Autres classes de complexité

Il ne s'agit pas d'établir un panorama exhaustif mais de présenter quelques classes qui définissent une plus grande granularité dans l'approche des propriétés du PCM.

Hiérarchie polynomiale

La hiérarchie polynomiale (**HP**) consiste en une construction de classes de complexité incluant les classes **P**, **NP** et **co-NP** précédemment définies. La définition de ces classes consiste en une sorte de généralisation du passage de la classe P à la classe NP par la notion d'oracle. Initialement proposée par [Stockmeyer, 1976], elle est définie récursivement :

Définition 2.37
Cas de base : $\mathbf{\Delta_0^P = \Sigma_0^P = \Pi_0^P = P}$
Pour tout entier $k \geq 0$:
$\mathbf{\Delta_{k+1}^P = P^{\Sigma_k^P}} \quad \mathbf{\Sigma_{k+1}^P = NP^{\Sigma_k^P}} \quad \mathbf{\Pi_{k+1}^P = co - NP^{\Sigma_k^P}}$

[Stockmeyer and Meyer, 1973] montrent une relation entre l'alternance des quantificateurs existentiels $\exists$ et universels $\forall$ pour une formule *QBF - Quantified Boolean Formula*. Il est possible de montrer que **HP** $\subseteq$ **PSPACE** mais la principale question est de savoir **HP** $\stackrel{?}{=}$ **PSPACE**

Hiérarchie FPT

La théorie de la NP-Complétude apporte une première idée sur la difficulté a priori d'un problème mais il est forcé de constater l'intérêt d'affiner et d'établir de nouvelles propriétés au sein des problèmes NP-complet. Nous

rappelons que pour chaque problème de décision nos algorithmes apportent une réponse positive ou négative et par extension une solution aux questions posées. De plus nous associons dans une majorité des cas un paramètre qui précise l'état de solution que nous devons calculer. Ce paramètre dénoté k peut faire référence par exemple à une taille de la solution. Nous redéfinissons un langage de décision auquel nous adjoignons un paramètre.

Définition 2.38 (Langage paramétré)
Un langage paramétré $\mathcal{L} \subseteq \Sigma^* \times \mathbb{N}$, où Σ désigne un langage fini et $\mathbb{N}$ l'ensemble des entiers naturels. Si nous considérons, $\langle x, k \rangle \in \mathcal{L}$, k est alors le paramètre associé aux données x.

Définition 2.39 (Fixed-Parameter Tractable - FPT))
Un problème P est *tractable*, avec paramètre fixé si l'appartenance de $\langle x, k \rangle \in \mathcal{L}_P$ est décidé en un temps au plus de $f(k)|x|^c$, avec c comme constante et une fonction $f : \mathbb{N} \mapsto \mathbb{N}$.

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 [Dinur and Safra, 2004, Karakostas, 2005]. Comme 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 2.40 (FPT-réduction)
$\mathcal{L}$ se réduit à $\mathcal{L}'$ par une réduction $\preccurlyeq_{fpt}$ et noté $\mathcal{L} \preccurlyeq_{fpt} \mathcal{L}'$, 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 $\langle \lambda(x), f(k) \rangle \in \mathcal{L}'$. Généralement, les fonctions f et f' sont équivalentes ce qui amène la FPT-réduction à préserver le paramètre ($f(k) = k$).

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 [Downey and Fellows, 1995] de la forme suivante :

$$FPT \subseteq W[1] \subseteq W[2] \subseteq ... \subseteq XP$$

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 précise 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. Particulièrement, le résultat $FPT \neq W[1]$ indique que la classe FPT est donc confinée [Bodlaender et al., 2009].

NP-optimisation et approximation

Il est évident qu'il existe une relation étroite entre un problème d'optimisation Π_o et le problème de décision Π_d associé. Il suffit d'une Turing-réduction faisant appel à un oracle pour obtenir notre problème reformulé sous forme d'un problème de décision. Immédiatement nous pouvons conclure que tout problème d'optimisation Π_o est NP-difficile si le problème de décision Π_d est NP-complet. Pour davantage de résultats, nous pouvons citer les travaux de [Escoffier, 2005].

Dans la cadre de la résolution non-déterministe en temps polynomiale nous pouvons de même définir la classe des problèmes d'optimisation **NPO**.

Définition 2.41 (classe NPO)
Nous appelons un problème de NP optimisation tout problème d'optimisation tel que :

- Le langage de $\mathcal{I}$ est dans **P** ;
- Il existe une fonction polynôme p telle que pour toute instance $i \in \mathcal{I}$ et pour toute solution réalisable $s \in \mathcal{S}$, nous avons la relation $|x| \leq p(|s|)$;
- Savoir si une solution x est réalisable est dans **P** ;
- La fonction d'évaluation f est calculable en temps polynomial.

Cette classe est introduite pour contenir les problèmes d'optimisation, en particulier les problèmes combinatoires qui nous intéresserons par la suite, dont le problème de décision est dans **NP**.

Proposition 2.5
Si Π est un problème de **NPO**, alors la version décision Π_d de Π est dans **NP**.

La caractérisation des problèmes **NPO** par la définition précédente et le résultat sur la NP-complétude, nous permettent maintenant de conclure sans surprise sur la difficulté des problèmes.

Corolaire 2.1
Si $\mathbf{P} \neq \mathbf{NP}$, il est impossible de résoudre un problème de la classe **NPO** en un temps polynomial si sa version de décision est NP-complet.

L'impossibilité de résoudre efficacement les problèmes **NPO** dans le pire des cas, nous amène à repenser la résolution de ces problèmes d'optimisations [Papadimitriou and Yannakakis, 1991]. Une solution envisagée est de ne calculer qu'une solution approchée par rapport à la solution optimale en garantissant tout de même un rapport de qualité. Un choix s'opère sur l'efficacité de l'algorithme qui devient polynomial au détriment de la garantie d'exhiber la meilleure solution. Historiquement dans [Johnson, 1974], l'auteur définit le rapport d'approximation pour les problèmes d'optimisation entre la valeur de la solution courante et celle de l'optimum.

Définition 2.42 (Rapport d'approximation)
Pour tout problème d'optimisation Π_o de **NPO**, le rapport d'approximation est défini par $\rho_{\Pi_0}(i, s) = \frac{val(s)}{Opt(i)}$ pour toute solution admissible s de i.

Le rapport d'approximation permet une mesure relative de la qualité de la solution trouvée par rapport à celle de la solution optimale. Pour le cas d'un problème de minimisation ce rapport est défini en $[1; +\infty[$, et $[0; 1]$ pour un problème de maximisation. Dans les deux cas, l'approximation est d'autant meilleure que ce rapport tend vers 1. La question se pose évidemment sur l'existence d'un algorithme A dont l'approximation est calculable pour n'importe quelle instance du problème d'optimisation. On dit alors que A est ρ-approché si, pour toute instance i, le rapport d'approximation de $A(I)$ est meilleur que $\rho(I)$.

Définition 2.43 (ρ-approximable)
Un problème d'optimisation Π_o est dit approximable à rapport ρ (ou ρ-approximable) s'il existe un algorithme polynomial ρ-approché pour Π_0.

Nous pouvons d'ores et déjà opérer une première distinction entre le problème d'optimisation. Une première propriété intéressante sur ρ concerne une approximation dont le ratio est constant et ne dépendant donc plus de la taille en entrée. Une telle propriété peut-être vue comme un pourcentage sur la qualité que nous pouvons espérer pour la résolution d'une instance quelconque.

Définition 2.44 (Classe APX)
La classe **APX** désigne l'ensemble des problème de NP-optimisation qui admet un algorithme d'approximation polynomiale ρ constant.

2.4 Conclusion

Dans ce chapitre, nous avons présenté une introduction à la logique propositionnelle pour fournir un cadre à la résolution du PCM. Le formalisme des fonctions booléennes et plus particulièrement des pdBfs nous permettra par la suite de définir une modélisation qui respectera le plus possible les attentes des biologistes. Cette résolution nous a amené à considérer le PCM comme un problème d'optimisation combinatoire. Pour résoudre ces problèmes, nous avons présenté des techniques basées d'une part sur des méthodes complètes qui nous permettent de garantir une solution optimale mais peuvent demander un effort calculatoire très important et des ressources en terme de temps et d'autre part, les méthodes incomplètes qui proposent un compromis entre la qualité de la solution et le temps d'exécution. Enfin, nous avons présenté quelques éléments sur la théorie de la complexité avec des résultats récents qui nous permettra par la suite de nous donner une référence et des précisions sur la difficulté des approches développées dans cette thèse.

3

Quelques approches pour l'apprentissage et la construction de formules booléennes

Sommaire

3.1 Apprentissage automatique 63
3.2 Apprentissage supervisé 64
3.3 Sélection d'attributs 65
3.3.1 Méthodes de régression 67
3.3.2 SVM 68
3.4 Minimisation de formule DNF 70
3.5 Conclusion 71

Tel que nous l'avons décrit, le problème de caractérisation multiple peut s'apparenter à un problème de construction de formules booléennes à partir de tables de vérités partielles (voir chapitre 2 pour plus de précisions) mais également à un problème d'apprentissage plus classique puisqu'il s'agit *in fine* d'identifier l'appartenance d'un nouvel individu (échantillon) à un groupe. De ce point de vue très général, il est clair que de nombreuses approches ont été proposées pour répondre à cette question. Nous allons donc, dans ce chapitre, sans vouloir toutefois en proposer un inventaire exhaus-

tif, rappeler brièvement certaines de ces approches qui nous semblent les plus pertinentes. Nous entendons par là qu'elles auraient pu constituer des candidates idéales pour le problème que nous abordons dans cette thèse. Notons que ces approches ont d'ailleurs été expérimentées afin d'en observer les limites face aux spécificités de notre problème, en particulier de sa complexité algorithmique. Une discussion synthétique est proposée tout au long des sections suivantes afin d'étayer le fait que ces méthodes usuelles ne peuvent pleinement répondre au problème de caractérisation multiple.

Nous plaçons la description des méthodes dans le cheminement de pensée qui a conduit les travaux de cette thèse, permettant ainsi d'identifier les pistes alternatives qui ont été envisagées.

Le problème pratique qui nous intéresse a été complètement décrit sous son aspect biologique dans le chapitre 1. Revenons sur la genèse du problème de caractérisation multiple de données : identifier l'appartenance d'un individu à un groupe en minimisant le nombre de caractères qui nous permettront de le discriminer par rapport à d'autres groupes possibles.

De ce point de vue, des liens évidents apparaissent avec l'apprentissage automatique dont l'objectif est d'apprendre à partir de données de base, l'appartenance de nouveaux individus à telle ou telle classe. Avant d'aller plus en avant dans nos investigations des méthodes d'apprentissage automatique, nous pouvons rappeler les bases fondamentales de ce domaine vaste, au cœur de l'intelligence artificielle, mais dont les frontières communes avec les statistiques et les mathématiques appliquées sont évidentes.

3.1 Apprentissage automatique

Leslie Valiant [Valiant, 1984] introduit la théorie du *probably approximately correct learning - (PAC learning)* qui définit un cadre mathématique pour les processus d'apprentissage dans le contexte booléen. La méthode décrite repose sur un ensemble d'échantillons dont il faut extraire la description d'une classe du concept à apprendre. Ainsi pour tout problème, il faut pouvoir garantir que ce qui est appris est proche du concept visé initialement.

Nous définissons plus formellement ce cadre par :

Définition 3.1 (Notation)

- X est l'ensemble des exemples et $\mathcal{C} = \{+1, -1\}$ leurs classes ;
- H une classe de concepts définis sur X ;
- D une distribution sur X ;
- La probabilité d'erreur d'une hypothèse $h \in H$ visant un concept $c \in H$ est $err(h) = p(c(x) = h(x))\ x \in D$;
- $O(c, D)$ est un oracle qui tire un exemple de X suivant D et le fournit avec sa classification par le concept cible :$\langle x, c(x) \rangle$.

Dans la cadre de la PAC, nous considérons qu'il est possible d'apprendre un modèle en un temps raisonnable tout en garantissant une marge d'erreur.

Définition 3.2 (PAC-apprenable)
H est PAC-apprenable s'il existe un algorithme A de paramètre $\epsilon > 0$ et $0 < \delta < 1$ qui fournit en temps polynomial dépendant de $(\frac{1}{\epsilon}, \frac{1}{\delta})$ une classification pour h, de probabilité $1 - \delta$, avec une erreur $err(h) < \epsilon < \frac{1}{2}$.

Dans la continuité, l'apprentissage de fonctions booléennes à partir d'exemples a été intensivement étudié dans [Hellerstein and Servedio, 2007, Natarajan, 1987, Gavaldà and Thérien, 2009]. 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 caractérisation (minimale) parmi un ensemble, plutôt qu'une fonction cible fixée. Toutefois, les travaux les plus proches sont certainement ceux sur l'apprentissage de k-juntas [Pensa et al., 2006]

mais la généralisation à plusieurs classes n'est pas traité. Ces auteurs apportent aussi des résultats sur la PAC-apprenabilité pour l'apprentissage de fonctions booléennes en essayant de trouver une formule booléenne.

Partant de ces fondements théoriques de nombreuses approches abordent l'apprentissage automatique sous des angles différents. De retour à notre problème de caractérisation, une approche possible aurait pu consister à tenter de classer automatiquement les entités en groupes homogènes en utilisant des algorithmes de classification supervisée.

3.2 Apprentissage supervisé

Les algorithmes d'apprentissage statistique supervisé consistent en général à regrouper à priori des items (étiquetage -*labeling*) en classes d'éléments proches, permettant ensuite d'attribuer une classe à un nouvel élément sur la base de sa proximité avec l'une des classes existantes. Notons, qu'il existe bien évidemment des méthodes non supervisées [Bailey et al., 2002, Pensa et al., 2006] pour lesquelles ni le nombre de classes ou ni la répartition des items ne sont pas connus a priori.

Concernant la classification non supervisée qui consiste à construire k classes d'individus cohérents, il existe naturellement un grand nombre d'algorithmes dont le célèbre k-means [Tou and González, 1974]. Ce type d'approche nécessite la définition d'une métrique (par exemple en ce qui nous concerne la distance de Hamming entre deux entités), ce qui constitue l'un des points durs de la performance de ces algorithmes. Ensuite, les items sont progressivement regroupés afin de minimiser la distance intra-classe (par rapport au centroïde de celle-ci). Le processus est ainsi itéré jusqu'à stabilité des classes. Un tel algorithme pourrait être mis en œuvre pour constituer des groupes de pathovars homogènes, toutefois, il faut souligner que notre problématique n'est pas une problématique de classification non supervisée puisque la connaissance des groupes est supposée initialement certaine et exhaustive.

En effet, les données sont fournies par des experts. 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. L'intérêt des méthodes classiques de classification comme les réseaux bayésiens devient moins pertinent. En fait, nous voulons apprendre une représentation plus compacte mais exacte des données fournies par l'utilisateur.

Mentionnons également que des approches de reconstruction d'arbres phylogénétiques ont largement été étudiées, notamment basées sur des approches de classification hiérarchique (par exemple, UPGMA [Michener and Sokal, 1957]) ou des heuristiques [Goëffon, 2006]. Toutefois, comme rappelé au chapitre 1, la phylogénie du genre *Xanthmonas* fait débat et se baser sur une telle approche pour caractériser les groupes n'était pas pertinent du point de vue biologique.

Maintenant que nous avons écarté certaines approches, nous pouvons revenir sur notre problème de caractérisation avec comme entrée nos groupes de pathovars. Une première question naturelle qui se pose alors est de savoir s'il ne serait pas envisageable de réduire le nombre de caractères à prendre en compte initialement. La sélection d'attributs est un problème très classique en apprentissage pour réduire la dimension de l'espace des objets à apprendre (cela peut intuitivement être perçu dans notre problème comme la dimension en largeur alors que le nombre d'entités présentes dans la matrice initiale correspond à une dimension verticale, ces deux dimensions ayant bien évidemment un impact sur la difficulté de l'instance à traiter).

3.3 Sélection d'attributs

Nous n'allons pas ici dresser une liste exhaustive des méthodes existantes mais plutôt rappeler les principes généraux de cette problématique.

La sélection d'attributs est largement utilisée en biologie pour déterminer, lors de l'observation d'un phénomène, quels sont les facteurs sans influence ou peu influents ou encore les facteurs redondants. Cela permet naturellement de réduire la dimension des entrées afin de réduire le coût de l'apprentissage. En particulier, une telle démarche peut à la fois permettre d'améliorer les performances d'une classification ou de mieux interpréter le résultat de l'apprentissage, en identifiant aussi quels sont les facteurs les plus importants.

L'objectif de la sélection d'attributs est donc d'identifier le sous-ensemble d'attributs de taille minimale nécessaire et suffisant pour définir le concept cible, ici la caractérisation multiple. En pratique, il s'agit de sélectionner un sous-ensemble d'attributs de taille $n < d$, tel qu'un critère soit optimisé par rapport à tous les sous-ensembles de taille n.

Il faut alors définir la notion de pertinence d'un attribut, ce qui n'est pas simple et dépend en général du problème d'apprentissage que l'on souhaite résoudre.

Par exemple, dans notre cas, un caractère serait pertinent si toutes les entités d'un groupe ont la même valeur pour ce caractère et toutes les enti-

tés des autres groupes avaient une valeur différente pour ce caractère. Dans ce cas il est clair que le caractère serait à lui seul discriminant pour le groupe considéré. Toutefois, cette situation est sans doute plutôt rare. Nous pouvons alors utiliser des mesures d'entropie des caractères pour en estimer la distribution des valeurs par rapport aux groupes et en inférer leur pertinence.

De manière générale la sélection d'attributs est un problème de nature combinatoire NP-difficile et repose en pratique sur des approches heuristiques. Le schéma suivant extrait de [Dash and Liu, 1997] synthétise la démarche qui est en général suivie.

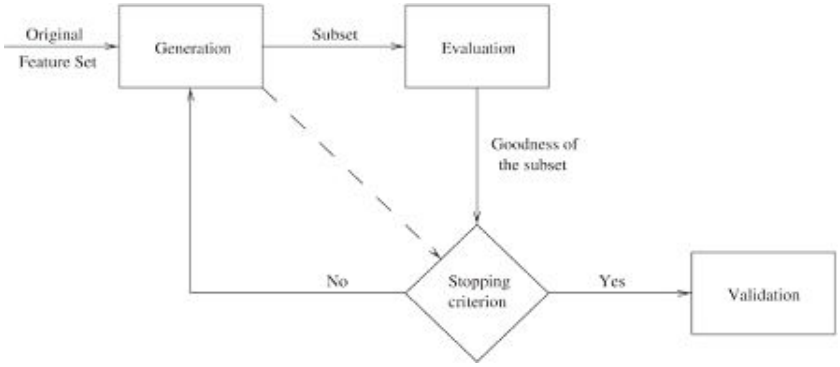


Figure 3.1 : Sélection d'attributs

Nous pouvons ainsi situer notre problème par rapport aux travaux relatifs aux *feature extraction* [Guyon and Elisseeff, 2003]. 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).

Maintenant que nous venons d'évoquer une possibilité pour réduire la dimension du problème. Nous pouvons nous pencher sur la caractérisation des entités d'un groupe par rapport aux autres groupes. Immédiatement, nous pourrions lier cette problématique à la notion d'apprentissage supervisé binaire et de classifieurs binaires. Parmi, les diverses approches nous pouvons en particulier considérer les séparateurs à vastes marges (SVM). L'objectif serait donc d'apprendre un classifieur qui nous permettrait de séparer les entités d'un groupe de toutes les autres entités.

Dans un premier temps nous présentons les méthodes de régressions qui sont utilisées dans le choix de gènes.

3.3.1 Méthodes de régression

Les méthodes de régression sont des méthodes statistiques dont le but est de connaître une relation entre des variables quantitatives pour effectuer généralement une prédiction. En statistique le terme « quantitatif » désigne le fait que les variables sont mesurables et dont les opérations arithmétiques ont un sens et s'oppose au terme « qualitatif » ou « modale ». Par exemple une variable correspondant à une observation sur le poids ou l'âge fait parti de la première catégorie mais une variable indiquant le genre d'un individu appartient à la deuxième.

Définition 3.3 (Modèle de régression classique)

- n : designe le nombre d'individus ;
- p : le nombre de variables observées ou explicatives ;
- y_i : la valeur du caractère étudié pour l'individu i ;
- x_{ij} : la valeur du $j^{\text{ème}}$ variable de l'individu i ;
- e_i : est le résidu. Les résidus sont gaussiens et indépendants ;
- β_i : les paramètres que nous devons estimer. Une estimation de ce vecteur est noté $\hat{\beta}$.

Un modèle de régression lineaire est :

$$y_i = \sum_{j=1}^{p} \beta_j x_{ij} + \beta_0 + e_i$$

que nous écrivons généralement :

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{e}$$

Nous notons β désigne le vecteur des paramètres que nous devons estimer en prenant compte de e.

Généralement, la méthode des moindres carrés est utilisée pour calculer $\hat{\beta}_{BOS}$ (Basic Ordinary Square).

$$\hat{\beta}_{BOS} = (X^t X)^{-1} X^t y$$

Dans le cadre du choix d'un problème de variables nous introduisons le vecteur γ. La sélection s'opère pour la variable $j^{\text{ème}}$ variable en ayant $\gamma_j = 1$. Pour $\gamma_j = 0$ la variable n'est pas sélectionnée.

$$\mathbf{y} = \mathbf{X}\beta\gamma + \mathbf{e}$$

Particulièrement pour la sélection de gènes nous observons que $p << n$ ce qui demande le développement de nouvelles techniques basées par exemple sur les régression pénalisées comme LASSO [Tibshirani, 1994] et plus récemment ElasticNet [Zou and Hastie, 2005].

3.3.2 SVM

Les SVM (*support vector machine* - séparateur à vaste marge) [Cortes and Vapnik, 1995, Meyer et al., 2003] sont des outils développés dans le cadre de de la classification binaire à partir des travaux sur les réseaux de neurones. L'idée est de chercher un hyperplan pouvant séparer un ensemble d'échantillons distribués entre deux classes. Cet hyperplan est calculé pour maximiser la marge c'est à dire la distance entre cet hyperplan et les observations.

Définition 3.4 (SVM)
Pour n échantillons pour $k = 1, \ldots, n$:

- x_k : désigne un vecteur d'observation ;
- l_k : représente l'étiquetage du vecteur x_k dans les classes $C = \{+1, -1\}$;
- w : désigne le vecteur poids qu'on cherche à calculer.

Il s'agit alors

$$minimiser\ \frac{1}{2}\|w\|^2$$

sous les contraintes

$$l_k(w^t x_k + w_0) \geq 1$$

Visuellement, la figure 3.2, issue de [Cornuéjols and Zucker, 2010], illustre l'objectif général de cette approche.

Bien évidemment, de nombreux problèmes ne sont pas linéairement séparables. La technique consiste alors à effectuer un changement d'espace et de dimension via la notion de fonction noyau figure 3.3. Nous renvoyons le lecteur à [Cornuéjols and Miclet, 2010] pour plus de précisions. Intuitivement, le schéma montre une transformation Φ permettant de plonger le problème initial dans un cadre linéaire. Ces transformations classiques peuvent par

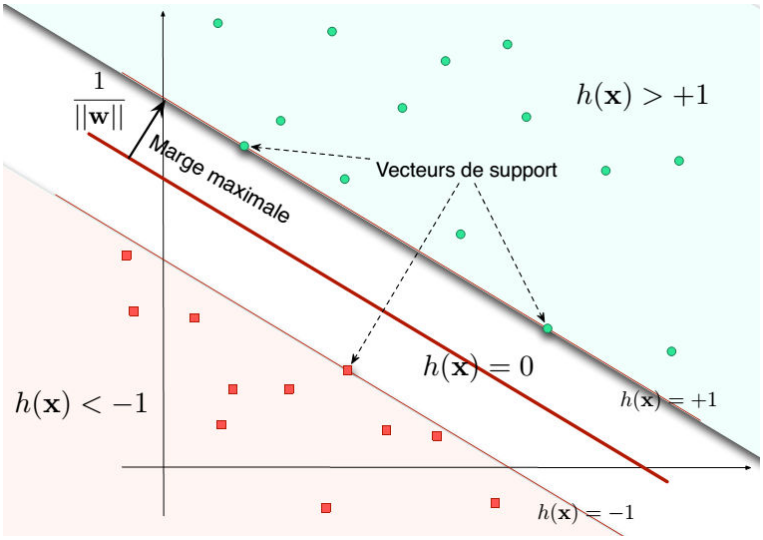


Figure 3.2 : Support Vector Machine

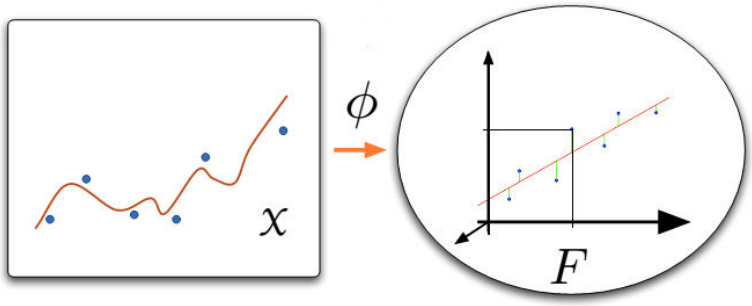


Figure 3.3 : Méthodes à noyau

exemple utiliser des fonctions non-linéaires quadratiques. La non-linéarité se retrouve ainsi traitée dans la transformation elle-même.

Toutefois, il se peut dans certains cas que les observations ne puissent être séparées de manière linéaire. Une première solution consiste aussi à introduire des variables ressorts ξ_k pour obtenir une tolérance pour nos marges. Nous pouvons maintenant poser le système suivant où C désigne la tolérance admise pour notre modèle :

$$minimiser \ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_k$$

sous les contraintes

$$l_k(w^t x_k + w_0) \geq 1 - \xi_k$$

Un SVM ne permet pas de garantir une classification valide à 100%. Rappelons que notre objectif n'est pas d'apprendre une généralisation à partir d'exemples mais de synthétiser la connaissance de manière exhaustive. Les SVM ont donc été écartés. De plus, en raison du très grand nombre de caractères à prendre en compte, il ne semblait pas raisonnable d'utiliser cette approche de manière combinée sur tous les groupes bien que ce formalisme s'étende à des problèmes à plusieurs classes [Chapelle and Keerthi, 2008, Obozinski et al., 2006].

Enfin, l'objectif est que la solution obtenue puisse apporter une information intelligible aux utilisateurs biologistes, ce qui ne serait pas le cas dans le cadre d'une approche de type SVM où le classifieur résultant ne peut être exploité dans ce sens.

Nous concentrant à présent sur une vision différente de notre problème, nous considérerons par la suite que la caractérisation d'un groupe peut être vue comme la construction d'une formule de caractérisation définie en logique propositionnelle. Les données en entrées étant considérées comme la définition d'une table de vérité partielle des différentes formules à construire (voir chapitre 2, cette problématique se rapproche alors des travaux effectués dans ce sens en électronique).

3.4 Minimisation de formule DNF

Dans le domaine de l'électronique, il est intéressant lors de la conception (*design*) de circuits, de réduire le nombre de portes logiques. Il s'agit d'implémenter toute fonction booléenne avec le circuit minimal équivalent. Particulièrement, pour les *FPGA*, qui les implémentent sous forme de DNF. Les tableaux de Karnaugh [Karnaugh, 1953] sont une première méthode dont le principe se base par une simplification (figure 3.1).

En 1956, Quine et McKluskey ([McCluskey, 1956]) utilisent les impliquants premiers pour obtenir une minimisation d'une fonction et montrent le caractère NP-difficile de ce problème. Actuellement plusieurs logiciels existent, dont Expresso basé sur la recherche locale [Brayton et al., 1984]. Christopher Umans établit que le problème de minimisation d'une formule DNF est Σ_2^p-complet [Umans, 2001]. Naïvement, la minimisation d'une formule booléenne consiste à calculer un sous-ensemble d'impliquants premiers, puis à reconstruire une DNF équivalente à la première [Rymon, 1994].

Exemple 3.1
Soient x_1, x_2 et x_3 des variables propositionnelles et f une fonction booléenne à 3 variables. Une simplication de la la formule par la méthode de

Karnaugh est $f(x_1, x_2, x_3) = \neg x_3 \vee (x_1 \wedge \neg x_2)$

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
F	F	F	V
F	F	V	F
F	V	F	V
F	V	V	F
V	F	F	V
V	F	V	V
V	V	F	V
V	V	V	F

La minimisation de DNF s'applique aussi à d'autres domaines comme les systèmes experts pour les diagnostics en vue d'améliorer des moteurs d'inférences [de Kleer et al., 1992, de Kleer, 1999].

3.5 Conclusion

Nous avons dans ce chapitre, présenté de manière sélective un ensemble de méthodes que nous avons envisagées dans notre démarche de résolution de ce problème pratique de caractérisation. Les méthodes issues de la littérature ne répondent que partiellement aux objectifs posés. Les méthodes d'apprentissage supervisé sont avant tout des méthodes de classification. Le principe qui leur est associé, se focalise sur l'élaboration d'un classifieur (modèle logique ou mathématique) que l'on essaye de généraliser dans le but de prédire, en tenant compte des problèmes liés aux sur ou sous apprentissage. Dès lors, un apprentissage à 100% (sans erreur) n'a plus de sens pour ces méthodes. Ainsi, nous sommes plus proche de la compilation de connaissances ([Marquis, 2008]). Hormis cette première difficulté, nous mentionnons que les formules booléennes utilisées par les méthodes basées sur la logique inductive répondent simplement et complètement à la question de la représentation de nos solutions.

En conjonction, les travaux menés dans le cadre de la minimisation de circuits logiques électroniques ont permis d'établir des résultats théoriques précis en terme de complexité. De ces résultats, nous avons appréhendé la difficulté du PCM.

Ainsi, la multitude des méthodes nous a amené vers une synthèse, combinant ces divers formalismes afin d'exprimer correctement le problème de caractérisation multiple. Cette nouvelle approche est présentée dans les chapitres suivants.

II

Contribution au problème de caractérisation multiple

4

Formalisation du problème de caractérisation multiples (PCM)

Sommaire

4.1 Contexte du PCM **76**
4.2 Définitions et propositions **78**
4.3 Résolution par les fonctions booléennes partiellement définies (pdBfs) **82**
4.4 NP-Complétude et FPT-Hiérarchie **84**
4.5 Autres propriétés **87**
4.5.1 Bornes et satisfiabilité 87
4.5.2 Monotonie et langages d'extension 87
4.6 Conclusion **89**

Nous présentons dans ce chapitre la modélisation du PCM qui a évolué au cours de cette thèse.

4.1 Contexte du PCM

Comme nous l'avons décrit dans le chapitre 1, le problème de caractérisation multiple consiste à identifier de manière sûre des individus d'un groupe en fonction de certains de leurs caractères mesurables. Pour cela, nous proposons de construire une formule logique utilisant ces caractères comme des variables booléennes. Cette formule devra être satisfaite par les individus (entités) représentatifs du groupe et falsifiée par les entités des autres groupes. Une telle formule doit être ainsi construite pour chaque groupe. Pour des raisons de coût et de fiabilité, nous avons également vu que le nombre total de caractères utilisés par ces formules devait être minimal. Nous illustrons ce problème sur un exemple simple.

Dans la figure 4.1, nous considérons 5 entités réparties dans 3 groupes. La description de chaque entité se base sur un ensemble de 3 caractères, mesurés par une valeur booléenne à (0/1, présence/absence, ...). 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. Dans l'exemple de la figure 4.1, le groupe 1 est caractérisé par la négation simultanée des variables a et b et donc la formule $\neg a \wedge \neg b$. Le groupe 2 est caractérisé par les variables b et c avec la formule $b \wedge c$. 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 (c'est-à-dire discriminer ses entités par rapport aux autres groupes).

Plus concrètement, par rapport aux notions introduites dans les chapitres précédents, 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.

Le problème se base sur une structuration des données précises fournies par les biologistes :

- Le domaine des données : l'absence ou la présence des caractères nous amène à définir un domaine qui soit binaire (booléen). La sémantique

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 4.1 : Exemple de problème de caractérisation multiple

naturellement retenue est 0 ou faux dénote l'absence d'un caractère et 1 ou vrai caractérise la présence.

- Les entités : une entité est totalement définie sur un ensemble de caractères dont nous ne mettons pas en cause la véracité sur l'expression de ses caractères.

- Les groupes : un groupe est constitué par un nombre fini d'entités. L'ensemble des groupes est lui aussi considéré comme correct dans le sens où chaque entité appartient à un seul groupe.

Ainsi, par rapport à l'ensemble des méthodes proposées pour l'analyse génétique l'aspect du bruit et particulièrement le problème biais/variance n'est pas considéré. Nous écartons dans notre modélisation le caractère prédictif (généralisation du classifieur) qui n'est pas pris en compte. La modélisation du problème de caractérisation doit garantir plusieurs aspects :

- Les solutions doivent être informellement représentées par des « combinaisons » sur les caractères. Ces combinaisons nous ont amené à considérer les formules propositionnelles construites sur un langage fixé à partir de fonctions booléennes.

- Les formules doivent « séparer » sans ambiguïté et sans erreur.

- La modélisation doit prendre l'aspect minimal de la solution.

Comme nous l'avons déjà évoqué, notre objectif dans cette thèse ne sera pas de calculer des modèles d'une formule propositionnelle mais plutôt de construire des formules qui satisfont un modèle donné. Concrètement, comme nous venons de le voir nous associons des tables de vérité aux différents connecteurs qui en définissent ainsi l'interprétation. De la même manière, la problématique qui va nous concerner consistera à construire des

formules dont la table de vérité est donnée. Nous voyons donc que, d'un point de vue opérationnel, notre propos est complètement dual par rapport au problème classique de résolution.

4.2 Définitions et propositions

Définition 4.1 (Instance du PCM)
Une instance du PCM est définie par un n-uplet $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ où $\mathcal{X}$ est l'ensemble des variables propositionnelles, $\mathcal{E}$ est l'ensemble des entités définies sur $\mathcal{X}$ et l'ensemble des groupes $\mathcal{G} \subseteq 2^{\mathcal{E}}$.

Afin de modéliser ce problème, nous devons établir le lien entre la formulation ci-dessus et les éléments de logique propositionnelle que nous avons rappelés dans le chapitre 2.

- Chaque entité représente une affectation booléenne, ou interprétation, définie comme $e : \mathcal{X} \to \{0, 1\}$, où 0 et 1 sont respectivement les valeurs de vérité fausse et vraie. $e(x)$ correspond 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 valeurs 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 dont les n colonnes correspondent aux variables booléennes 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 l'instance initiale en réduisant le nombre de variables réellement utiles et/ou le nombre d'entités. Les informations redondantes peuvent être stockées pour guider la recherche ou reconstruire à posteriori de nouvelles solutions Nous définissons ainsi la notion d'instance non redondante.

Définition 4.2 (Instance non redondante)
Une instance est non redondante ssi :

- Il n'existe pas de colonne dont toutes valeurs sont identiques : $\nexists j \in \{1, \ldots, n\}, \forall i \in \{1, \ldots, m\}$, $e_{ij} = 1$ (resp $e_{ij} = 0$) ;

- Chaque colonne est unique : $\nexists j \in \{1,\ldots,n\}, \forall k \in \{1,\ldots,n\} \setminus \{j\}, \forall i \in \{1,\ldots,m\}$, $a_{ij} = a_{ik}$;
- Chaque entité est unique :
  $\nexists i \in \{1,\ldots,m\}, \forall l \in \{1,\ldots,m\} \setminus \{i\}, \forall j \in \{1,\ldots,n\}$, $e_i(x_j) = e_l(x_j)$.

L'application d'un algorithme d'élimination de la redondance s'effectue au pire des cas en $\mathcal{O}(|\mathcal{X}|^2 + |\mathcal{E}|^2)$ 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 4.3 (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 4.4 (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 $\forall i \in \{1,\ldots,|\mathcal{G}|\}, g_i \in \mathcal{G}$, $g_i \models \phi_i$.

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

Définition 4.5 (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. $\exists g \in \mathcal{G}, Sol(g) = \emptyset$).

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 $\phi_e^+ \equiv \bigwedge_{x \in \mathcal{X}} \delta(e, x)$ où $\delta : \mathcal{E} \times \mathcal{X} \to \mathcal{L}$ 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 $\phi_e^- \equiv \bigvee_{x \in \mathcal{X}} \neg\delta(e,x)$ et $\phi_g^- \equiv \bigwedge_{g' \in \mathcal{G} \setminus \{g\}} \bigwedge_{e' \in g'} \phi_{e'}^-$ (qui est en CNF). Nous notons $\phi_g \equiv \phi_g^+ \wedge \phi_g^-$.

Proposition 4.1 (Satifisfiabilité d'une instance)
Soit une instance $\mathcal{I} = (\mathcal{X}, \mathcal{E}, \mathcal{G})$, nous avons :

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

Preuve 4.1
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 l'ensemble $\{\phi_g\}_{g \in \mathcal{G}}$ qui est une solution de $\mathcal{I}$ puisque chaque formule ϕ_g caractérise g (définition 4.3). Donc $SOL(\mathcal{I}) \neq \emptyset$.

$L \Rightarrow R$:
Nous devons prouver que $\neg R \Rightarrow SOL(\mathcal{I}) = \emptyset$. Considérons $\neg R$:

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

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

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

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

Nous avons besoin du lemme suivant.

Lemme 4.1
$\forall e, e' \in \mathcal{E}, e \models \bigwedge_{x \in \mathcal{G}} \delta(e', x) \Leftrightarrow e = e'$

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

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

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

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

Nous avons $\neg R \Rightarrow SOL(\mathcal{I}) = \emptyset$.

Corolaire 4.1
Soit $\mathcal{I} = (\mathcal{X}, \mathcal{E}, \mathcal{G})$:

$$\exists g, g' \in \mathcal{G}, g \neq g', \exists e \in g, \exists e' \in g', e = e' \Leftrightarrow SOL(\mathcal{I}) = \emptyset$$

Remarquons que le corollaire précédent prouve que si $\mathcal{G}$ 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 :

Corolaire 4.2
Soit $\mathcal{I} = (\mathcal{X}, \mathcal{E}, \mathcal{G})$:

$$\mathcal{I}\ est\ satisfiable \Leftrightarrow \forall g, g' \in \mathcal{G}, g \neq g', g \cap g' = \emptyset$$

Nous exhibons alors une solution canonique $\Phi_{Max}^{\mathcal{I}}$ pour n'importe quel instance satisfiable $\mathcal{I}$:

$$\Phi_{Max}^{\mathcal{I}} = (\phi_{g_1}, \ldots, \phi_{g_{|\mathcal{G}|}})$$

Cette solution canonique est aussi une solution maximale.

Proposition 4.2
Soit $\mathcal{I} = (\mathcal{X}, \mathcal{E}, \mathcal{G})$, nous avons les propriétés suivantes :

1. $SOL(\mathcal{I}) \neq \emptyset \Leftrightarrow \mathcal{G} \models \Phi_{Max}^{\mathcal{I}}$
2. $\forall \Phi \in SOL(\mathcal{I}), \Phi \leftrightarrow \Phi_{Max}^{\mathcal{I}}$

3. $\forall \Phi \in SOL(\mathcal{I}), |\Phi^{\mathcal{I}}_{Max}| \geq |\Phi|$

Il est maintenant évident que toute instance possédant deux interprétations identiques dans deux groupes différents n'est pas satisfiable. Une conséquence immédiate : une instance non-redondante est satisfiable puisque chaque entité a une interprétation unique.

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.

La taille des solutions du PCM est définie par le suite :

Définition 4.6 (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 de ϕ.

Exemple 4.1 (formule)

- $a \wedge b \vee (c \wedge b)$ est une formule positive de longueur 4 et de taille 3.
- $a \wedge b \wedge \neg c$ est une CNF unitaire de taille et de longueur 3.

Définition 4.7 (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 4.8 (MINPCM (problème d'optimisation))
Pour une instance $\mathcal{I}$, une caractérisation optimale multiple minimale est un ensemble de formules $\Phi^* \in Sol(\mathcal{I})$ vérifiant $|\Phi^*| \leq |\Phi|$ avec $\forall \Phi \in Sol(\mathcal{I})$

4.3 Résolution par 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 avons donc généralisé le formalisme des pdBfs (voir chapitre 2) à plusieurs classes.

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. Nous redéfinissons pour chaque groupe une $pdBf_g$ qui est définie comme une pdBf avec $\mathcal{C}^+ = g$ et $\mathcal{C}^- = \mathcal{G} \setminus \{g\}$. Une instance du PCM est maintenant vue comme $pdBf_g, g \in \mathcal{G}$.

Définition 4.9 (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 $\mathcal{X}'$.

L'intérêt est de se concentrer sur les variables candidates de la solution optimale. Soit k la dimension de la projection. π_{pdBf_g} est la projection associée à la $pdBf_g$. Nous notons $g \models \pi$ (c-à-d, π_{pdBf_g} est consistante) la propriété :

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

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

Proposition 4.3 (Projetabilité et satisfiabilité)
$\mathcal{I} = (\mathcal{X}, \mathcal{E}, \mathcal{G})$ est satisfiable ssi $\mathcal{I}$ est projetable sur $\mathcal{G}$, c'est-à-dire $\forall g \in \mathcal{G}, \exists \phi, g \models \phi \Leftrightarrow \exists \pi, g \models \pi$

Preuve 4.2

$L \Rightarrow R$: 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 $\forall e \in g, e(x_i) = 1$ (resp. $e(x_i) = 0$) et $\forall e' \in \mathcal{G} \setminus \{g\}, e'(x_i) = 0$ (resp. $e(x_i) = 1$). 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 soit $\phi' = \phi \wedge x_i$, soit $\phi' = \phi \vee x_i$, telle que $g \models \phi'$ (selon le principe du cas de base). Nous avons $\exists \pi', g \models \pi'$.

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

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 groupes.

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.

4.4 NP-Complétude et FPT-Hiérarchie

Le SETCOVER est un problème défini comme étant un des problèmes de couverture et il appartient aux 21 problèmes de Karp avec entre autres SAT. Une couverture d'un problème est la solution d'un sous-ensemble d'éléments qui permettent par une propriété spécifique d'atteindre l'ensemble initial des éléments. Dans la littérature le problème de SETCOVER pour les graphes est associé bien souvent au problème de DOMINATINGSET. Il s'agit de trouver alors un sous-ensemble de sommets d'un graphe connecté à tous les sommets du graphes.

Définition 4.10 (SETCOVER)
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 $\bigcup_{c \in \mathcal{C}'} = \mathcal{U}$.

Définition 4.11 (MINSETCOVER)
Étant donnée une collection $\mathcal{C}$ de sous-ensembles d'un ensemble $\mathcal{U}$ nous cherchons s'il existe une sous-collection $\mathcal{C}' \subseteq \mathcal{C}$ vérifiant $\bigcup_{c \in \mathcal{C}'} = \mathcal{U}$ et tel que $|\mathcal{C}'|$ soit minimale.

Exemple 4.2
Soit un ensemble $\mathcal{U} = \{1, 2, 3, 4, 5\}$ et une collection $\mathcal{C} = \{\{3, 4\}, \{1, 2, 4\}, \{1, 4, 5\}, \{2, 3, 5\}\}$. Une couverture pour $k = 2$ est $\mathcal{C}' = \{\{1, 4, 5\}, \{2, 3, 5\}\}$ (qui est aussi une solution du MINSETCOVER) .

Nous allons utiliser la FPT-réduction et le fait que le problème SETCOVER soit de la classe de complexité paramétrée W[2] pour obtenir la complexité du PCM. Par la suite nous considérerons implicitement les problèmes de décisions.

Proposition 4.4
PCM et SETCOVER sont équivalents par FPT-réduction.

Preuve 4.3
Rappelons tout d'abord la définition du problème SETCOVER paramétré :

Montrons tout d'abord PCM $\preccurlyeq_{fpt}$ SETCOVER.

Soit $\langle \mathcal{I}, k\rangle = \langle (\mathcal{X}, \mathcal{E}, \mathcal{C}), k\rangle$ 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 associé 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 $g, g' \in \mathcal{G}$:

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

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

Soit π, une projection de dimension k, solution d'une instance $\mathcal{I} = (\mathcal{X}, \mathcal{E}, \mathcal{G})$, 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 $\mathcal{C}' = \bigcup_{x \in \mathcal{U}} c_x$ est bien une solution de l'instance $\langle (\mathcal{U}, \mathcal{C}), k\rangle$ de SETCOVER 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 $\mathcal{U}$. 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 vérifiant $e(x) \neq e'(x)$. 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 SETCOVER $\preccurlyeq_{fpt}$ PCM.

Soit $\langle (\mathcal{U}, \mathcal{C}), k\rangle$ une instance de SETCOVER 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

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

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

La réduction depuis SETCOVER n'utilisant que deux groupes, nous obtenons le résultat suivant.

Corolaire 4.3
PCM est W[2]-difficile.

La difficulté tient même si l'on se restreint aux instances contenant seulement deux groupes. Nous pouvons conclure de ce fait que PCM est W[2]-complet au vu de [Downey and Fellows, 1999]. 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 $\mathcal{O}(\binom{|\mathcal{X}|}{k})$. 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 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.

Nous réutiliserons le principe de cette réduction dans la section 5.3 du MINPCM vers le problème MINONES afin d'utiliser un solveur en programmation linéaire en nombre entier. Nous retrouvons un résultat similaire sur la difficulté similaire (non FPT) pour problème [Kratsch and Wahlström, 2010].

4.5 Autres propriétés

4.5.1 Bornes et satisfiabilité

Nous pouvons établir une borne inférieure et supérieure pour notre problème. La borne inférieure (BI) correspond au nombre minimal de variables nécessaire pour caractériser tous les groupes. Si nous connaissons le nombre minimal requis pour caractériser chaque groupe, il est évident qu'il n'y pas de plus petit nombre de variable qui autorise la caractérisation de tous les groupes. En ce qui concerne la borne supérieure (BS), nous pouvons être certain qu'il existe une solution pour l'ensemble des groupes avec, au plus, autant de variables que nécessaire pour caractériser chaque groupe.

$BI = |min(SOL(\mathcal{I}))| \geq max(max(|min(Sol(g_1))|, \ldots, |min(Sol(g_n)|), \lceil log_2(|\mathcal{G}|\rceil))$
et
$BS = |min(SOL(\mathcal{I}))| \leq |min(min(Sol(g_1)), \ldots, min(Sol(g_n)))|$

4.5.2 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 (chapitre 2). 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 4.12 (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 4.13 (Extension de $\preceq$ sur les entités)
L'ordre partiel $\trianglelefteq$ est l'extension naturelle sur les multi-ensembles 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.

Proposition 4.5 (monotonie de la satisfiabilité par $\trianglelefteq$ pour $\mathcal{F}^+$)
Soient e et e', deux entités et ϕ_+ une formule positive telle que $e \models \phi_+$. Si $e \trianglelefteq e'$ alors $e' \models \phi_+$.

Nous devons donc montrer 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 $\trianglelefteq$.

Preuve 4.4
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 que les entités minimales de la classe C^+ ne soient pas dominées par les entités (maximale) 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.

4.6 Conclusion

Par la formalisation du PCM, le choix de la logique booléenne s'est naturellement imposé par son pouvoir d'expressivité et sa simplicité. Dès lors, les nombreux échanges avec les biologistes ont permis d'affiner cette modélisation et de clairement définir ce que nous avons dénommé la « caractérisation multiple » et qui est une caractéristique dominante de notre problème. Lors de cette première phase, une première série d'algorithmes a été développée et sera présentée par la suite à la section 5.1.

L'utilisation des pdBfs a été induite par la nécessité d'introduire un mécanisme global pour satisfaire le critère d'optimalité et d'ancrer le PCM dans la littérature (sélection d'attribut) modulo l'introduction d'une étape intermédiaire relative au choix des variables de la solution finale. Une fois cette difficulté levée, nous avons proposé une série d'algorithmes à la section 5.2.

Enfin nous avons développé une étude montrant que le problème de décision associé au PCM se révèle difficile à résoudre même avec les méthodes les plus récentes. La W[2]-complétude de ce problème ne nous donne malheureusement pas d'accès à toutes les techniques disponibles pour la résolution de problèmes FPT [Guo and Niedermeier, 2007, Flum and Grohe, 2006] et nous oriente vers l'amélioration de l'heuristique du choix de variable. Dans [Creignou et al., 2001], les auteurs montrent plusieurs résultats théoriques sur les variantes de MINONES par rapport à la syntaxe des clauses. Dans le cas général MINONES est un problème APX-difficile, mais dans le cas présenté par la suite où les variables apparaissent toutes positives, MINONES est APX-complet. Nous retrouvons un résultat similaire pour le problème du SETCOVER dans [Cygan et al., 2009]. Ces résultats sont une piste de recherche sur la difficulté du MINPCM.

5

Présentation des algorithmes

Sommaire

5.1 Approche par générations de formules 93
5.1.1 Quilan au premier ordre Prolog - EXACTFORMCAR 93
5.1.2 Algorithme génétique 95
5.1.3 Limites de ces approches « directes » 97
5.2 Approche par les pdBfs 97
5.2.1 Heuristique et voisinage de recherche 97
5.2.2 Recherche complète 98
5.2.3 Recherche locale 99
5.3 Reformulation du PCM en programmation linéaire 99
5.3.1 Règles de recodage 100
5.3.2 Simplifications d'instance 101
5.4 Conclusion 102

Dans ce chapitre, nous exposons les techniques mises en œuvre pour la résolution du problème de caractérisation. Nous avons adapté et développé plusieurs méthodes basées sur le principe de la figure 5.1. Nous distinguons tout d'abord les méthodes « directes ». Nous exposons dans une seconde partie les méthodes « indirectes » basées sur les pdBFs.

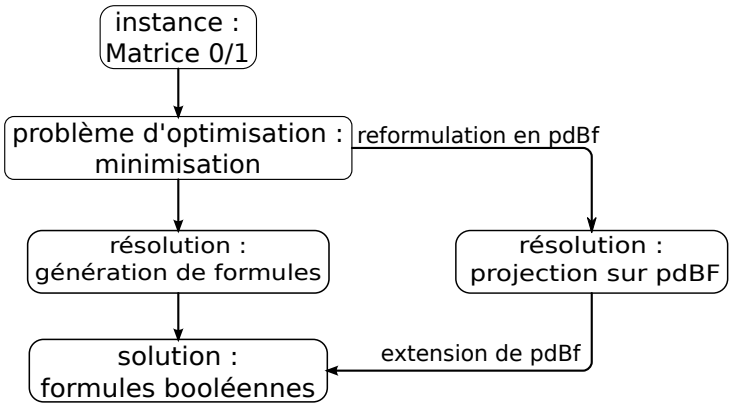


Figure 5.1 : Présentation générale des algorithmes développés durant cette thèse

5.1 Approche par générations de formules

Nous avons dans un premier temps développé des méthodes basées sur la génération d'ensembles de formules, en essayant de respecter les contraintes du PCM. Cette approche dite « directe » permet de fournir un ensemble de formules booléennes.

5.1.1 Quilan au premier ordre Prolog - EXACT-FORMCAR

Ross Quilan propose un algorithme *(FOIL - First Order Inductive Learner)* d'apprentissage de clauses de Horn en logique du premier ordre [Quinlan, 1990]. Il s'agit d'apprendre une formule, par le biais de prédicats, en temps polynomial en essayant de séparer les exemples positifs des exemples négatifs. La construction est inductive et utilise une heuristique qui consiste à satisfaire le plus grand nombre d'exemples positifs, en réfutant les négatifs. Dans le cadre d'une restriction vers la logique propositionnelle, l'algorithme de FOIL est chargé de construire une formule sous de HORN.

Nous délaissons l'approche multiple du PCM au profit d'une caractérisation d'un groupe versus tous les autres. Généralement une heuristique de calcul d'entropie est utilisée pour le choix de la variable possédant la valeur portant le plus d'informations, en particulier le choix se porte sur la variable pouvant valider le plus d'entités d'un groupe. Les expérimentations préliminaires nous ont indiqué que la formule trouvée possédait bien souvent l'ensemble de toutes les variables et se rapprochait alors de la taille de la formule canonique c'est pourquoi nous avons donc élargi le champ des formules apprises en implémentant un générateur se basant sur la construction récursive et exhaustive des formules CNF.

Il est évident que passer en revue toutes les formules n'est pas efficace et devient non polynomial. En effet, beaucoup d'entre elles sont équivalentes mais il est compliqué d'éviter de manipuler des formules redondantes car la génération de CNF sans redondance est difficile et s'apparente au problème de détection de clauses redondantes qui est coNP-complet [Darras et al., 2005, Fourdrinoy et al., 2007]. La commutativité des connecteurs logiques est une première cause d'équivalence. Par conséquent, en ordonnant les clauses, nous pouvons réduire de manière conséquente le nombre de distributions possibles des littéraux dans les clauses. Nous utilisons une notion de motif de clauses qui correspond aux distributions possibles des variables dans les clauses.

Exemple 5.1
Les motifs possibles pour une CNF utilisant deux occurrences de littéraux sont :

- $()_1 \wedge ()_1$: 2 clauses avec 1 occurrence chacune.
- $()_2$: 1 clause avec 2 occurrences.

On pourra alors instancier ces motifs avec des variables.

Exemple 5.2
A partir des motifs de l'exemple précédent et en considérant une formule utilisant 2 occurrences de littéraux avec 2 variables x et y, nous pouvons construire les CNF suivantes :

- $()_1 \wedge ()_1$: $(x) \wedge (y)$, $(\neg x) \wedge (y)$, $(x) \wedge (\neg y)$ et $(\neg x) \wedge (\neg y)$
- $()_2$: $(x \vee y)$, $(\neg x \vee y)$, $(x \vee \neg y)$ et $(\neg x \vee \neg y)$

De manière identique aux clauses, les littéraux sont eux aussi ordonnés dans chaque clause. Durant l'instanciation des motifs, un mécanisme de simplification vérifie si les littéraux n'apparaissent pas dans des clauses unitaires. Dans ce cas, la CNF est équivalente à une autre CNF de taille plus petite par subsomption.

Il est intéressant de remarquer que le nombre de motifs est donné par la fonction de partition d'un entier, notée $p(n)$. Pour un entier n, $p(n)$ retourne le nombre de façons distinctes (ne dépendant pas de l'ordre des termes) de représenter n comme une somme d'entiers naturels. Une approximation asymptotique de cette fonction a été proposée par Hardy et Ramanujan [Hardy and Ramanujan, 1918], et Rademacher [Rademacher, 1937] propose la formule 5.2 qui peut être utilisée pour évaluer le nombre de formules examinées par EXACT-FORM-CAR.

Exemple 5.3
La fonction de partition $p(3) = 3$ car :

- $3 = 1 + 1 + 1$
- $3 = 1 + 2$
- $3 = 0 + 3$

Nous avons aussi testé d'autres fragments intéressants du langage de la logique propositionnelle en restreignant aux formules des classes $\mathcal{F}^{\wedge}$ et $\mathcal{F}^{\wedge}_{+}$. Dans ce cas nous pouvons écarter tout problème de redondance.

$$p(n) = \frac{1}{\pi\sqrt{2}} \sum_{k=1}^{\infty} A_k(n)\sqrt{k}\frac{d}{dn}\left(\frac{\sinh\left(\frac{\pi}{k}\sqrt{\frac{2}{3}\left(n-\frac{1}{24}\right)}\right)}{\sqrt{n-\frac{1}{24}}}\right)$$

avec

$$A_k(n) = \sum_{0\leq h\leq k,(h,k)=1} \mathrm{e}^{\pi \mathrm{i}\, s(h,k)-2\pi \mathrm{i} nh/k}$$

$s(h,k)$ désigne la somme de Dedekind et (h,k) indique que h et k sont premiers entre eux.

Figure 5.2 : formule de fonction de partition d'un entier

5.1.2 Algorithme génétique

Dans [Chhel et al., 2012] nous avons proposé un algorithme évolutionnaire AGFORMCAR, pour essayer de contrôler la taille d'un ensemble de solutions en introduisant une pénalité sur la qualité de caractérisation de chaque formule en fonction de la taille de la solution finale du PCM. La population est composée par un ensemble d'individus représentant une formule.

Algorithme 4 : Algorithme évolutionnaire pour le PCM basé sur l'approche directe

Données : Une instance $(\mathcal{X}, \mathcal{E}, \mathcal{G})$
Résultat : L'ensemble des meilleurs individus P^*
$P \leftarrow$ InitPopulation();
tant que *not StopCondition* **faire**
  $\phi_p \leftarrow RandomIndividual(P)$;
  $\phi_p \leftarrow Mutation(\phi_p)$;
  $(p_1, p_2) \leftarrow SelectParents(P)$;
  $e \leftarrow Crossover(p_1, p_2)$;
  $e \leftarrow Mutation(e)$;
  $P \leftarrow Update(P, e)$;
  $P^* \leftarrow UpdateBest(P, P^*)$;
fin

Pour une raison d'implémention et de conception, les individus sont des formules CNF. Nous définissions la taille de chaque individu par la taille de chaque formule.

L'initialisation de la population P consiste à générer des individus distincts dont la taille est limitée par le paramètre $\phi_{\max}$. Si $\phi_{\max} = 1$, chaque

individus de départ est une formule composée d'un seul littéral. Les expérimentations que nous avons menées ont retenu cette valeur et avec $|P| = 30$ nous obtenons des résultats satisfaisants.

À chaque itération l'algorithme se déroule :

1. Tout d'abord un individu est choisi aléatoirement sur lequel est appliqué un des sept opérateurs de mutation dont le rôle est de diversifier les formules ou de modifier la taille de celle-ci. Cette première mutation permet que AGFORMCAR ne soit pas élitiste :
   - *flip* : change le signe d'un littéral dans la formule ;
   - *flip+* : change le signe de toutes les occurrences d'un littéral dans la formule ;
   - *add* : ajoute un littéral à une clause ;
   - *del* : supprime un littéral d'une clause ;
   - *delv* : supprime une variable de la formule ;
   - *addc* : ajoute une clause aléatoire ;
   - *delc* : supprime une clause.
2. Ensuite, deux parents sont sélectionnés par un tournois en 3 tours (voir [Eiben and Smith, 2003] pour plus de détails sur les stratégies usuelles d'un algorithme évolutionnaire) et la formule est générée à partir d'un opérateur de croisement mono-point. À ce point une autre mutation aléatoire peut survenir. La descendance est insérée dans la population tandis que les plus mauvais individus sont retirés.
3. (P^*) est mis à jour et contient l'ensemble des meilleurs individus.

Soit une instance $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ et un groupe $g \in G$ à caractériser. Nous évaluons la qualité des individus par le nombre d'entités qu'ils caractérisent et elle est calculée comme étant la moyenne pondérée de deux fonctions :

1. $f^+(\phi)$: Le nombre d'entités g, qui ne sont pas caractérisées parϕ.
2. $f^-(\phi)$: Le nombre d'entités g $G \setminus \{g\}$, caractérisées par ϕ.

Si $f^+(\phi) = f^-(\phi) = 0 = f(\phi)$, alors ϕ caractérise g.

L'algorithme se termine quand la condition d'arrêt est obtenue. Une condition est le nombre d'itérations (ici le nombre de croisements). Si nous sommes juste intéressés à trouver une solution alors l'algorithme s'arrête lorsque les formules sont toutes caractérisantes. Sinon le processus continue en essayant de minimiser le nombre de variables au respect de la définition de MINPCM.

5.1.3 Limites de ces approches « directes »

En utilisant cette première approche directe nous sommes confrontés à trois difficultés. À cause des définitions récursives de la logique propositionnelle, nous pouvons construire des termes (formules) de taille infinie. En effet, il est possible de construire la formule $f = a \wedge a \wedge a \wedge \ldots$. Nous pouvons toutefois limiter leur taille (qui croit exponentiellement avec le nombre de variables) en ne travaillant qu'avec certaines formes normales (DNF ou CNF par exemple).

Beaucoup de formules sont équivalentes mais il est difficile d'éviter de manipuler des formules équivalentes car la génération de CNF sans redondance revient à résoudre le problème de détection de clauses redondantes qui est coNP-complet. La commutativité des connecteurs logiques est une première cause d'équivalence. Par conséquent, en ordonnant les clauses, nous pouvons réduire de manière conséquente le nombre de distributions possibles des littéraux dans les clauses.

Le problème majeur est de raisonner globalement sur $\mathcal{I}$ pour trouver $min(SOL(\mathcal{I}))$. Trouver une formule pour chaque groupe avec une contrainte globale (en minimisant le nombre total de variables) est très coûteux. Nous avons besoin d'un mécanisme global pour résoudre le problème.

5.2 Approche par les pdBfs

Cette modélisation s'est avérée la plus adaptée au PCM.

5.2.1 Heuristique et voisinage de recherche

Une étude attentive des problèmes de couverture nous amène à considérer que le PCM ne nécessite que deux voisinages de base et un troisième facultatif. Soit $\mathcal{X}$ l'ensemble des variables de notre problème, ainsi une solution s est un sous-ensemble non vide de $\mathcal{X}$.

- Voisinage add - $\mathcal{V}_a$: Ce voisinage permet d'acquérir une nouvelle variable dans la solution courante. $(s, s') \in \mathcal{V}_a$ ssi $s = \{x_1, \ldots, x_n\}$ et $s' = \{x_1, \ldots, x_n, x_{n+1}\}$. Dès lors $|s'| = |s| + 1$ et ce voisinage ne peut être utilisé quand s contient l'ensemble des caractères.

- Voisinage delete - $\mathcal{V}_d$: Ce voisinage permet de retirer une variable de la solution courante. $(s, s') \in \mathcal{V}_d$ ssi $s = \{x_1, \ldots, x_{n-1}, x_n\}$ et $s' = \{x_1, \ldots, x_{n-1}\}$. La conséquence est que $|s'| = |s| - 1$ et ce voisinage ne peut être utilisé quand s contient seulement un seul caractère.

- Voisinage exchange - $\mathcal{V}_e$: $(s, s') \in \mathcal{V}_d$ ssi $s = \{x_1, \ldots, x_i, \ldots, x_n\}$ et $s' = \{x_1, \ldots, x_{i'}, \ldots, x_{n-1}\}$ avec $x_i \neq x_{i'}$. Ce voisinage permet d'échanger une variable de la solution courante par une autre variable absente. Par conséquence $|s'| = |s|$ et ce voisinage n'a pas à être utilisé quand s contient l'ensemble des caractères.

5.2.2 Recherche complète

Le solveur EXACTPROJCAR, proposé dans [Chhel et al., 2012], 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.

- Un Branch and Bound dont les fonctions de calculs des bornes sont basées sur le nombre de variables de la fonction en cours. La borne minimale est initialisée à $k = \lceil log_2(|\mathcal{G}|) \rceil$, tandis que la borne supérieure est égale au cardinal de l'ensemble des solutions. À chaque fois qu'une solution s est trouvée $|s|$ devient la nouvelle borne maximale.

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 [Descamps and Véron, 1981] qui privilégie les variables permettant de séparer un groupe par rapport aux autres.

5.2.3 Recherche locale

RLPROJCAR est un algorithme basé sur une méthode de recherche locale. 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.

Algorithme 5 : Algorithme RLPROJCAR

Données : s une solution initiale,f une fonction objectif et un ensemble de voisinages $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$
Résultat : s^* une solution optimale locale
$s^* \leftarrow s$;
$k' \leftarrow 1$;
tant que $k' \leq k$ **faire**
  $s \leftarrow \text{Selection}(\mathcal{V}_{k'}(s^*))$;
  si $f(s^*) > f(s)$ **alors**
    $s^* \leftarrow s$;
    $k' \leftarrow 1$;
  fin
fin

5.3 Reformulation du PCM en programmation linéaire

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

La réduction vers MINONES reprend l'idée exposée de la proposition 4.4. Nous construisons une formule booléenne avec chaque paire d'entités de groupes différents.

Définition 5.1 (MINONES)
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 variables à vraie dans cette affectation.

Le problème MINONES est souvent défini en WEIGTHED MINONES, version pondérée de MINONES, où chaque variable possède une pondération (poids). L'objectif est maintenant de minimiser la somme des pondérations des variables à vrai. Dans notre le cas du MINONES cette pondération est vue comme constante pour chaque variable.

Exemple 5.4
Soit $\mathcal{X}_\phi = \{x_1, x_2, x_3, x_4\}$ un ensemble de variables propositionnelles de ϕ. Une solution du MINONES pour $\phi = (x_1 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee x_3) \wedge (\neg x1 \vee \neg x_2 \vee x_4)$ est $x_2 = 1$ ou $x_3 = 1$. La valeur optimale de ϕ pour le problème de MINONES est donc de 1.

5.3.1 Règles de recodage

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 transformation d'une instance est bornée par $|\mathcal{E}|^2$.

Exemple 5.5 (Transformation vers MINONES 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 :

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

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

En imposant un ordre sur les entités avec la transformation proposée, nous pouvons constater qu'une même contrainte ne peut apparaître plusieurs fois si l'instance est non redondante.

5.3.2 Simplifications d'instance

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 [Boufkhad and Roussel, 2000] comme cela est 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 décisions d'une contrainte produite par la paire d'entités $\{e_i, e_j\}$, nous pouvons supprimer toutes contraintes 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 $\forall c_{\{e_k,e_l\}}$,avec $k \neq l$, nous avons $var(c_{\{e_k,e_l\}}) \subset var(c_{\{e_i,e_j\}})$. 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 une solution $y_1 = 1$ et $y_3 = 1$ 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. $c_{\{e2,e5\}}$).

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 qui suit que pour des instances venant de problèmes réels les conditions de subsomption sont souvent réunies.

5.4 Conclusion

Nous distinguons deux étapes importantes dans la résolution du PCM. La première phase de développement issue directement des définitions sur le PCM s'est révélée en pratique peu efficace. En effet, la gestion du critère de minimisation sur un ensemble de formules booléennes est complexe. Ainsi, dans un deuxième temps, l'extension des pdBfs à plusieurs classes nous a permis de lever cette difficulté. Celle-ci est décomposée par une recherche sur l'ensemble de variables candidates de la solution optimale qui s'apparente aux méthodes de sélection de variables que nous avons nommée projection puis nous appliquons le mécanisme d'extensions vers le langage spécifié sur chaque pdBfs. Nous rappelons que nous pouvons prétraiter chaque instance dans par exemple certains cas où les formules sont positives.

Les expérimentations menées par la suite reposent essentiellement sur cette deuxième phase et sont exposées dans le chapitre suivant.

6

Expérimentations et valorisation

Sommaire

6.1 Instances 104
6.1.1 Instances Réelles 104
6.1.2 Instances Aléatoires 104
6.2 Travaux préliminaires et problèmes connexes . 105
6.2.1 Coefficient de Capacité Diagnostique (CCD) . . 105
6.2.2 Caractérisations Positives et Canonicité 106
6.2.3 Caractérisations groupe par groupe 106
6.3 Expérimentations 107
6.3.1 Valorisation 109
6.4 Conclusion 109

Dans ce chapitre, nous exposons l'application des algorithmes par approches par pdBfs proposées dans la partie précédente sur des problèmes de caractérisation biologique. Toutefois, bien que le nombre de données fournies par les biologistes augmente exponentiellement, le nombre de jeux de tests sur lequel nous pouvons travailler est assez faible. Il est donc intéressant de proposer des instances aléatoires afin d'augmenter le nombre d'instances mais aussi de pouvoir définir ce qu'est une instance difficile.

6.1 Instances

Nous présentons les instances réelles et artificielles sur lesquelles nous avons menées nos expérimentations.

6.1.1 Instances Réelles

Les instances réelles (raphv, raphy, rarep, rch8 et rch10) sont tirées de problèmes de caractérisation provenant en partie de l'API BioMérieux basée sur des propriétés bio-chimiques de l'espèce Ralstonia et sur des expressions de gènes de virulences de l'espèce *Xanthomonas*. Ces propriétés biochimiques sont un ensemble de fonctions métaboliques de la bactérie comme la présence d'une activité enzymatique particulière.

6.1.2 Instances Aléatoires

Il est difficile d'évaluer la puissance d'un algorithme au travers d'expérimentations réalisées sur un faible nombre d'instances réelles. Les instances sont telles représentatives des toutes les instances possibles ? Peut-on tirer des conclusions sur un petit nombre d'expérimentations ? La construction des instances ne fausse-t-elle pas les résultats ? Pour éviter ces problème, nous avons défini plusieurs instances aléatoires. Nous testons ensuite nos méthodes de résolutions afin d'observer leurs comportements.

Pour générer une instance aléatoire, plusieurs approches sont possibles. La méthode que nous utilisons nécessite 4 valeurs : le nombre de groupes ng, le nombre de caractères nc, le nombre d'entités total ne et un coefficient de bruit cb compris entre 0 et 1. Les instances aléatoires sont ensuite générées en deux étapes :

- ne entités distinctes sont construites de manière aléatoire ;
- $(1 - cb)ne$ entités sont réparties de manière équitable entre tous les groupes ;
- $cb.ne$ entités sont affectées aléatoirement dans un groupe.

Cette méthode de construction nous permet de garantir un certain nombre d'entités par groupe. Ces instances sont notées dans la table 6.2 : s*graine-bruit*.

6.2 Travaux préliminaires et problèmes connexes

Certains des résultats proposés aux biologistes se sont avérés inexploitable. En effet, bien que les solutions que nous leur fournissions étaient correctes, le test de l'absence ou de la présence d'un caractère peut parfois être compliqué expérimentalement. Il a donc fallu adapter nos méthodes pour répondre à leurs demandes.

6.2.1 Coefficient de Capacité Diagnostique (CCD)

La méthodes des CDD (Coefficient de Capacité Diagnostique) a permis dans un premier temps de répondre très partiellement aux attentes des biologistes [Hajri et al., 2009]. Cette méthodes propose de séparer les groupes en calculant un coefficient pour chaque variable en se basant sur la théorie de Bayes et le calcul de l'entropie au sens de Shannon. Nous rappelons pour une variable discrète X composée de n symboles s_i dont la probabilité d'apparition est P_i et qui codés dans une base b, L'entropie H est définie par

$$H(X) = -\sum_{i=1}^{n} p_i \log_b p_i$$

Ainsi, dans le cas fréquent du codage binaire, nous utilisons donc le logarithme de base 2.

Nous rappelons succinctement la formule de Bayes [Bayes and Price, 1763], qui est un outil essayant de déterminer une distribution a posteriori en utilisant les probabilités conditionnelles. Ainsi soient deux événements A et B associés aux probabilités $p(A)$ et $p(B)$. D'autre part nous avons la probabilité conditionnelle de A sachant B que nous notons $p(A/B)$.

$$p(A/B) = \frac{p(B/A) \times p(A)}{p(B)}$$

À partir des travaux [Descamps and Véron, 1981], nous avons défini pour le PCM un CDD pour chaque variable qui permet par la suite d'établir un classement (appelé aussi méthode de ranking en feature selection). Le principe est de calculer un CDD pour chaque variable, indépendamment des autres, et de classer l'ensemble des scores par ordre décroissant.

Nous définissons de nouvelles fonctions

- $f_{intra}(x_i, v, g_j)$ désigne la fréquence (intra) de la variable x_i de valeur $v = \{0, 1\}$ pour le groupe g_j.
- $p(g_j)$ détermine la probabilité d'appartenir au groupe g_j.
- $f(x_i, v)$ détermine la la fréquence de la variable x_i de valeur $v = \{0, 1\}$ sur l'ensemble des entités.

Nous considérons maintenant la variable x_i.

$$cdd(x_i) = \sum_{j=1}^{|\mathcal{G}|} \sum_{v=0}^{1} p(g_j) \times f_{intra}(x_i, v, g_j) \times \log_2 \frac{f_{intra}(x_i, v, g_j)}{f(x_i, v)}$$

Dans le formule si dessous nous pouvons remarquer que ces valeurs sont normalisées dans l'intervalle $[0, 1]$. Par la suite la méthode des CDD est considéré comme l'heuristique de base pour la résolution du PCM.

6.2.2 Caractérisations Positives et Canonicité

D'un point de vue biologique, tester expérimentalement la présence d'un caractère est généralement simple. Par contre, il est beaucoup plus difficile d'affirmer l'absence d'un caractère. L'absence est donc vu comme un défaut de l'expression d'un gène. Une première requête des biologistes a donc été de modifier la méthode de recherche afin de ne proposer que des solutions contenant des littéraux positifs. Ainsi, leur tests devenaient beaucoup plus fiables.

Une deuxième requête fut d'établir des formules booléennes « simples à lire ». Si le caractère de la logique propositionnelle ne fut pas remis en cause, nous nous sommes orientés naturellement vers des solutions comportant que des formules sous forme canoniques. Après discutions, nous avons retenus que ces formules devait appartenir à l'ensemble $\mathcal{F}^{DNF}$. La restriction à l'ensemble $\mathcal{F}_+^\wedge$ venant en complément du premier souhait des biologistes n'est utilisée que sur quelques instances.

Dés lors nous avons du adapter nos algorithmes en prenant en compte ces deux contraintes. D'une part grâce aux propriétés énoncées dans le chapitre 5 et d'autre part en adaptant le mécanisme d'extension.

6.2.3 Caractérisations groupe par groupe

Certains caractères sont plus faciles que d'autres à étudier. Toutefois, il ne faut pas éliminer les caractères difficiles car il se peut qu'ils soient nécessaires pour trouver une solution. Il a donc été demandé de fournir pour

chaque groupe plusieurs caractérisations possibles. De plus, certains tests ne concernent qu'un groupe. Il est donc intéressant de chercher les caractérisations minimales pour un groupe par rapport aux autres (PCM à deux groupes).

Il est intéressant de remarquer qu'il est très rare de pouvoir reconstruire la solution minimale du PCM en se basant sur les ensembles des caractérisations minimales de chaque groupe. Nous avons testé cette hypothèse pour l'instance rch8.

Le tableau 6.1 rassemble pour chaque groupe la taille des solutions minimales c'est a dire, la taille de la solution qui permet de caractériser un groupe sans prendre en compte la dimension globale du PCM a proprement parlé et le nombre de solutions minimales observé pour ce groupe. Rappelons que ce calcul correspond à un calcul de borne maximale pour le problème du PCM décrit dans le chapitre 5.

Groupe	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Taille des solutions minimales	3	2	4	2	4	2	4	2	3	3	3	3	2	3	3	3	3	2	2	1	1
Nombre de solutions minimales	11	4	1	7	4	9	9	2	7	1	29	30	9	1	27	2	10	2	8	2	1

Tableau 6.1 : Résultats expérimentaux sur rch8

La taille de l'espace de recherche (le nombre de solutions à recombiner) correspond au produit du nombre de solutions minimales de chaque groupe. Ainsi pour rch8 ce nombre est de l'ordre $1,89 \times 10^{14}$ et la taille des solutions globales minimales recombinées est de 16. Pour note, l'espaces de recherche pour calculé le PCM de l'instance rch8 s'élève au maximum au nombre $2^{37} \simeq 1,37 \times 10^{11}$. pour une taille de 9 pour les solutions minimales. Ainsi, nous avons rapidement délaissé l'approche locale par groupe pour se concentrer directement sur la solution globale.

En pratique vu le coût très important pour calculer pour chaque groupe une solution minimale, nous avons utilisé l'algorithme de recherche locale (RLPC) pour calculer la borne maximale et La borne minimale correspond à $\lceil log_2(|\mathcal{G}|\rceil)$.

6.3 Expérimentations

Le tableau 6.2, présente plusieurs informations sur les instances ainsi que sur leur résolution par différents algorithmes. Les propriétés des instances sont données par les 4 premières colonnes. La colonne *contraintes* fournit le nombre de contraintes nécessaires pour reformuler chaque instance sous forme de programme linéaire (partie 5.3), 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 pour la résolution en programmation linéaire en utilisant CPLEX [1] sur le problème réduit (PL), pour l'exécution de EXACTPROJCAR (EPC) et enfin pour l'application de l'algorithme de recherche locale (RLPC). Le temps autorisé pour chaque exécution est limité à 10 minutes. Les résultats en gras indiquent que les solutions sont optimales.

Instances	Entités	Groupes	Caractères	Contraintes	r	Réduction	PL	EPC	RLPC
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**		
rch10	173	27	98	5908	34	983	**10**	15	14

Tableau 6.2 : Résultats expérimentaux pour le PCM

Les résultats obtenus montrent tout d'abord les limites de la transformation du problème en programmation linéaire. Dès que le nombre de caractères augmente, le nombre de contraintes explose et il n'est plus possible de traiter le problème. De plus, comme nous l'avons mentionné dans la section 5.3.2, les instances aléatoires ne pouvant pas être réduites, leur résolution était impossible. En effet, étant limité en mémoire (32Go sur une machine 64 bits), nous n'avons pu lire (charger en mémoire) les instances aléatoires (noté par « - ») avec le solveur CPLEX. Pour les instances réelles, le nombre de caractères n'étant pas très élevé et les entités de chaque groupe étant assez proches, la modélisation sous forme de programmation linéaire permet une forte réduction et donc une résolution assez efficace. EPC fournit tout le temps la meilleure solution sauf pour deux instances réelles. Pour 4 instances, nous sommes surs que la solution trouvée est optimale. La recherche locale (RLPC) produit de bons résultats mais atteint rarement la borne

[1] http://www.ibm.com/software/integration/optimization/cplex-optimizer/

optimale. Toutefois, la solution est obtenue très rapidement.

6.3.1 Valorisation

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* [Boureau et al., 2012]. 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 [2] permettant aux biologistes d'obtenir aisément une caractérisation de données correspondant au PCM.

6.4 Conclusion

Les résultats sur les instances réelles ont été très satisfaisants pour les biologistes. Les méthodes complètes qui utilisent les pdBfs se sont révélées plus efficace que les méthodes par génération de formules et répondent aux contraintes posés par le PCM (minimisation globale). Malheureusement, faute de données réelles nous n'avons pu tester nos algorithmes sur de larges instances structurées. Malgré ce manque, les expérimentations appliquées sur des instances aléatoires, nous ont permis de tester la relative robustesse de notre approche avec les pdBfs. En effet même si la solution optimale n'est pas trouvée on constate une forte réduction de la dimension du problème. Cependant, nous pensons que ces instances sont faciles et il reste encore à comprendre la difficulté d'une instance en fonction de ces paramètres. Dans l'ensemble, La recherche locale permet d'obtenir de bon résultats ce qui nous permet d'utiliser cette méthodes pour le calcul rapide d'une solution et surtout d'établir une borne supérieur pour les méthodes complètes.

[2] http://forge.info.univ-angers.fr/~gh/Idas/Ccd/mcps/

Conclusion générale

Contributions sur le PCM

Nous avons du dégager au fil de cette thèse un nouveau cadre de modélisation et de résolution qui s'inscrit dans le processus de conception d'une puce à ADN. Le problème de caractérisation multiple a permis de répondre aux questions posés par la détection de pathovars présents sur les semences. En amont, nous avons essayé de positionner nos travaux dans l'ensemble des techniques d'identification de gènes issus des travaux de sélection d'attributs. Les travaux sur la logique inductive, ont fourni un premier support pour le développement de méthodes mais face à la difficulté de l'ensemble des contraintes liées à la minimisation de la solution et la gestion d'un ensemble de groupe, nous avons développé une approche à deux phases basées sur les pdBfs.

En effet, La résolution du PCM par le formalisme des pdBfs étendues à plusieurs classes et du mécanisme d'extension permettant de calculer la solution finale, ont permis de fournir d'une part un ensemble de résultats théoriques et d'autres part plusieurs algorithmes exacts et approchés pour ce problème. Cependant ces travaux restent une exploration préliminaire du PCM et les perspectives de recherches restent nombreuses.

Perspectives de recherche

En continuité des travaux présentés dans cette thèse, deux points précis sont à développer :

- L'heuristique des CCD utilisée comme mesure, nous a permis dans un premier temps d'établir une stratégie pertinente dans l'exploration de l'espace de recherche. Mais comme abordé au chapitre 3, il existent de nombreuses méthodes pour la sélection d'attributs. Il peut être intéressant de coupler ces méthodes au vue d'améliorer la qualité du choix de variables présentes dans la projection sur les pdBfs. Ainsi,

une étude comparative devra être menée entre toutes ces méthodes et nous avons à concevoir un outil qui pourrait nous donner a priori une idée sur la difficulté d'une instance.

- En complément, il manque un résultat important sur APX-complétude (ou non) du PCM. En effet, sa grande similitude avec le SETCOVER laisse envisager un résultat positif. Dès lors, l'approche des métaheuristiques semble tout a fait justifiée pour résoudre le PCM du fait d'un ratio approximation important.

Nous avons dans cette thèse imposé la logique propositionnelle comme support de nos solutions. Hors dans la suite de la logique inductive des travaux ont été mené dans l'apprentissage de contraintes [Sebag and Rouveirol, 1996]. En extension, il peut être intéressant de s'intéresser aux travaux sur la compilation de données [Marquis, 2008] car généralement les biologistes ne sont pas satisfait avec une unique solution. Il s'agirait de compiler un ensemble de solutions et d'extraire certaines informations par un processus de requêtes.

Annexe

le schéma RNG des fichiers d'entrée

Nous avons défini un ensemble de grammaire pour l'utilisation de fichier XML en entrée et en sortie.

```
# racine des elements

start = element instance{
 attribute nom{xsd:string},
 attribute nbgroupes{xsd:unsignedInt},
 attribute nbattributs{xsd:unsignedInt},
 attribute nbentites{xsd:unsignedInt},
 # décription de l'instance
 element description{xsd:string}?,
 element donnees{
  element groupes{
   element groupe{
    attribute id{IdGroupe},
    attribute nom{xsd:string}
   }*
     },
     # correspond aux colonnes
     element attributs{
      element attribut{
        attribute id{IdAttribut},
    attribute nom{xsd:string},
    attribute poids{xsd:integer}
   }*
     },
     # correspond aux lignes
     element entites{
  element entite{
    attribute id{IdEntite},
    attribute ref{RefGroupe},
```

```
    attribute nom{xsd:string},
    # correspond aux cellules
    # ici on peut ommettre la reference si vides on conserve
    # les elements valuationPositives et valuationNegatives
    # si pas d'attributs pour plus de visibilite
    element valuationsPositives{
     attribute refs{RefsAttributs}?
    },
    element valuationsNegatives{
     attribute refs{RefsAttributs}?
    }
    }*
  }
 }
}

# Types simples

IdGroupe=xsd:ID{pattern='g[1-9][0-9]*'}
RefGroupe=xsd:IDREF{pattern='g[1-9][0-9]*'}
RefsGroupes=xsd:IDREFS{pattern='g[1-9][0-9]*(g[1-9][0-9]*)*'}
IdAttribut=xsd:ID{pattern='a[1-9][0-9]*'}
RefsAttributs=xsd:IDREFS{pattern='a[1-9][0-9]*(a[1-9][0-9]*)*'}
IdEntite=xsd:ID{pattern='g[1-9][0-9]*e[1-9][0-9]*'}
RefsEntites=xsd:IDREFS{
 pattern='g[1-9][0-9]*e[1-9][0-9]*(g[1-9][0-9]*e[1-9][0-9]*)*'}
pourcentage=xsd:unsignedInt{maxInclusive='100'}
```

le schéma RNG des fichiers de sortie

```
# racine des elements

start=element resultats
{
 element statistiques{
  #indique les elements redondants
  element redondance{
   attribute pourcentage{pourcentage},
   element entites{
    attribute refs{RefsEntites}
   }*
  },
  element impossibilites{
   element entites{
    attribute refs{RefsEntites}
   }*
  }
 }?,
 element caracterisations{
  element groupe{
   attribute ref{RefGroupe},
   (element solution{formule})*
  }+
  }?
}

# Type complex recursif

formule=element and{op}|element or{op}|element not{formule}
 |element attribut{attribute ref{RefAttribut}}
op=element op{formule},element op{formule}+

# Types simples

RefGroupe=xsd:IDREF{pattern='g[1-9][0-9]*'}
RefAttribut=xsd:IDREFS{pattern='a[1-9][0-9]*'}
RefsEntites=xsd:IDREFS{
 pattern='g[1-9][0-9]*e[1-9][0-9]*( g[1-9][0-9]*e[1-9][0-9]*)*'}
pourcentage=xsd:unsignedInt{maxInclusive='100'}
```

L'ensemble des options du solveur

```
algo:
Generic options:
  -v [ --version ]      print version string
  -h [ --help ]         produce help message
  --config arg          name of a file of a configuration.

Configuration:
  -g [ --id-group ] arg    set of group id (default ALL)
  -a [ --algo ] arg        algo
  -m [ --min ] arg         minimal bound (default log_2(|G|)
  -M [ --max ] arg         maximal bound (|id-group|/2 )
  -e [ --exhaustive ]      exhaustive (default desactivate)
  -w [ --window ] arg      window's size (default 1)
  -c [ --cut ]             cut (default desactivate)
  -t [ --iteration ] arg   number of iterations(default 1000)
  -p [ --pertubation ] arg pertubation in [0,1] (default 0.95)
  -r [ --restart ] arg     number of restart(default 2)
  -d [ --dac ]             dac reading instead xml
  -x [ --no-xml ]          xml output (default activate)
  -S [ --cs ]              cs (default desactivate)
  -s [ --ds ]              ds (default desactivate)
  -C [ --mcs ]             mcs (default desactivate)
  -i [ --input-file ] arg  input file
  -o [ --output-file ] arg output file
```

Contributions personnelles

Liste des communications

Revues

- 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. *Revue d'Intelligence Artificielle 27(4-5) 2013* : 649-668

- T. Boureau, M. Kerkoud, F. Chhel, G. Hunault, A. Darrasse, C. Brin, K. Durand, A. Hajri,S. Poussier, C. Manceau, F. Lardeux, F. Saubion, M.-A. Jacques : A multiplex-PCR assay for identification of the quarantine plant pathogen Xanthomonas axonopodis pv. phaseoli. *Journal of Microbiological Methods* : 2012.

Conférences internationales

- Fabien Chhel, Frédéric Lardeux, Adrien Goëffon, Frédéric Saubion : Minimum multiple characterization of biological data using partially defined boolean formulas. *Symposium On Applied Computing (SAC) 2012* : 1399-1405

- Fabien Chhel, Adrien Goëffon, Frédéric Lardeux, Frédéric Saubion, Gilles Hunault, Tristan Boureau : Experimental Approach for Bacterial Strains Characterization. *EvoBio 2011* : 135-1405

- Fabien Chhel, Adrien Goëffon, Antoine Lafosse, Frédéric Lardeux, Frédéric Saubion, Gilles Hunault, Tristan Boureau : The bacterial strains characterization problem. *Symposium On Applied Computing (SAC) 2011* : 108-109

Conférences nationales et séminaire

- Fabien Chhel, Frédéric Lardeux, Adrien Goëffon, Frédéric Saubion : Caractérisation multiple minimale utilisant les formules booléennes partiellement définies. *Journées de l'Intelligence Artificielle Fondamentale (JIAF) 2011*

- Séminaire à l'Université de Caen en Novembre 2011.

- Fabien Chhel, Frédéric Lardeux, Frédéric Saubion : Caractérisation de souche bactériennes à l'aide la de logique bactérienne. *Journées Francophones de Programmation par Contraintes (JFPC) 2010*

Valorisations

Nous avons déposé un brevet à l'INPI pour le test de diagnostic sur Xanthomonas axonopodis pv. phaseoli.

Tristan Boureau, Fabien Chhel, Gilles Hunault, Mohamed Kerkoud, Frédéric Lardeux, Charles Manceau, Stéphane Poussier, and Frédéric Saubion : Procédé de depistage de xanthomonas axonopodis pv. phasaeoli, INPI numéro FR 2970480 (A1), 2012

Autres travaux

J'ai aussi, en marge de ma thèse, participé aux travaux de modélisation et de conception d'un algorithme de recherche locale sur le problème de positionnement des miroirs pour le four solaire Thèmis situés dans les Pyrénées. Ces travaux avec des physiciens ont bénéficié d'une publication dans une revenue internationale du domaine.

Adrien Salomé, Fabien Chhel, Gilles Flamant, Alain Ferrière ,Frederik Thiery : Control of the flux distribution on a solar tower receiver using an optimized aiming point strategy : Application to THEMIS solar tower. *Journal of Solar Energy (94) 2013* : 352-366

Références bibliographiques

[Altschul et al., 1990] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3) :403–410. (cité page 18)

[Bailey et al., 2002] Bailey, J., Manoukian, T., and Ramamohanarao, K. (2002). Fast algorithms for mining emerging patterns. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD)*, page 39–50. Springer. (cité page 64)

[Bayes and Price, 1763] Bayes, M. and Price, M. (1763). An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, a. m. f. r. s. *Philosophical Transactions*, 53 :370–418. (cité page 105)

[Beasley, 1990] Beasley, J. E. (1990). OR-library : Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11) :1069–1072. (cité page 39)

[Bodlaender et al., 2009] Bodlaender, H. L., Downey, R. G., Fellows, M. R., and Hermelin, D. (2009). On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8) :423 – 434. (cité page 58)

[Bollig and Wegener, 1996] Bollig, B. and Wegener, I. (1996). Improving the variable ordering of obdds is np-complete. *IEEE Transactions on Computers*, 45(9) :993–1002. (cité page 37)

[Boufkhad and Roussel, 2000] Boufkhad, Y. and Roussel, O. (2000). Redundancy in random SAT formulas. In *National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence (AAAI)*, pages 273–278. AAAI Press / The MIT Press. (cité page 101)

[Boureau et al., 2012] Boureau, T., Chhel, F., Hunault, G., Kerkoud, M., Lardeux, F., Manceau, C., Poussier, S., and Saubion, F. (2012). Procédé de depistage de xanthomonas axonopodis pv. phasaeoli. Brevet INRA Depose INPI : FR 2970480 (A1). (cité page 109)

[Boureau et al., 2013] Boureau, T., Kerkoud, M., Chhel, F., Hunault, G., Darrasse, A., Brin, C., Durand, K., Hajri, A., Poussier, S., Manceau, C., Lardeux, F., Saubion, F., and Jacques, M. (2013). A multiplex-pcr assay for identification of the quarantine plant pathogen xanthomonas axonopodis pv. phaseoli. *Journal of Microbiological Methods*, 92(1) :42–50. (cité pages 13 et 23)

[Brace et al., 1990] Brace, K. S., Rudell, R. L., and Bryant, R. E. (1990). Efficient implementation of a bdd package. In *ACM/IEEE Design Automation Conference (DAC)*, pages 40–45. ACM. (cité page 36)

[Braun and Gō, 1985] Braun, W. and Gō, N. (1985). Calculation of protein conformations by proton-proton distance constraints : A new efficient algorithm. *Journal of Molecular Biology*, 186(3) :611 – 626. (cité page 18)

[Bräysy, 2003] Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4) :347–368. (cité page 48)

[Brayton et al., 1984] Brayton, R. K., Sangiovanni-Vincentelli, A. L., McMullen, C. T., and Hachtel, G. D. (1984). *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA. (cité page 70)

[Bryant, 1986] Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8) :677–691. (cité page 36)

[Bryant, 1992] Bryant, R. E. (1992). Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3) :293–318. (cité page 36)

[Chapelle and Keerthi, 2008] Chapelle, O. and Keerthi, S. S. (2008). Multiclass feature selection with support vector machines. In *Proceedings of the American statistical association*. (cité page 70)

[Chhel et al., 2012] Chhel, F., Lardeux, F., Goëffon, A., and Saubion, F. (2012). Minimum multiple characterization of biological data using par-

tially defined boolean formulas. In *Symposium On Applied Computing (SAC)*, page 1399–1405. (cité pages 95 et 98)

[Cook, 1971] Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Symposium On Theory of Computing (STOC)*, pages 151–158. ACM. (cité pages 41 et 53)

[Cornuéjols and Miclet, 2010] Cornuéjols, A. and Miclet, L. (2010). *Apprentissage artificiel - Concepts et algorithmes*. Eyrolles. (cité page 68)

[Cornuéjols and Zucker, 2010] Cornuéjols, A. and Zucker, J.-D. (2010). Apprentissage à partir de données bio et médicales. Ecole d'hiver "Apprentissage Statistique et Data Mining". (cité page 68)

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3) :273–297. (cité page 68)

[Creignou et al., 2001] Creignou, N., Madhu Sudan, N., and Sudan, M. (2001). *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM). (cité page 89)

[Cygan et al., 2009] Cygan, M., Kowalik, Ł., and Wykurz, M. (2009). Exponential-time approximation of weighted set cover. *Information Processing Letters*, 109(16) :957–961. (cité page 89)

[Darras et al., 2005] Darras, S., Dequen, G., Devendeville, L., Mazure, B., Ostrowski, R., and Sas, L. (2005). Using boolean constraint propagation for sub-clauses deduction. In *Principles and Practice of Constraint Programming (CP)*, page 757–761. Springer. (cité page 93)

[Darwin, 1872] Darwin, C. (1872). *The Descent of Man, and Selection in Relation to Sex*. Number vol. 1 in The Descent of Man, and Selection in Relation to Sex. D. Appleton. (cité page 49)

[Dash and Liu, 1997] Dash, M. and Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, 1(1-4) :131–156. (cité page 66)

[de Kleer, 1999] de Kleer, J. (1999). An improved incremental algorithm for generating prime implicates. In Levesque, H. and Pirri, F., editors, *Logical Foundations for Cognitive Agents*, Artificial Intelligence, pages 103–112. Springer. (cité page 71)

[de Kleer et al., 1992] de Kleer, J., Mackworth, A. K., and Reiter, R. (1992). Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2) :197–222. (cité page 71)

[Descamps and Véron, 1981] Descamps, P. and Véron, M. (1981). Une méthode de choix des caractères d'identification basée sur le théorème de bayes et la mesure de l'information. *Annales de Microbiologie (Paris)*, 132B. (cité pages 98 et 105)

[Dinur and Safra, 2004] Dinur, I. and Safra, S. (2004). On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162 :2005. (cité page 57)

[Downey and Fellows, 1999] Downey, R. and Fellows, M. (1999). *Parameterized complexity*. Monographs in Computer Science, Springer-Verlag. (cité page 86)

[Downey and Fellows, 1995] Downey, R. G. and Fellows, M. R. (1995). Fixed-parameter tractability and completeness i : Basic results. *SIAM Journal on Computing (SICOMP)*, 24(4) :873–921. (cité page 57)

[Eiben and Smith, 2003] Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer. (cité page 96)

[Eiter et al., 2002] Eiter, T., Ibaraki, T., and Makino, K. (2002). Decision lists and related boolean functions. *Theoretical Computer Science (TCS)*, 270 :493–524. (cité page 38)

[Engvall and Perlmann, 1972] Engvall, E. and Perlmann, P. (1972). Enzyme-linked immunosorbent assay, elisa iii. quantitation of specific antibodies by enzyme-labeled anti-immunoglobulin in antigen-coated tubes. *The Journal of Immunology*, 109(1) :129–135. (cité page 21)

[Escoffier, 2005] Escoffier, B. (2005). *Approximation polynomiale de problèmes d'optimisation : aspects structurels et opérationnels*. PhD thesis, Université Paris-Dauphine. (cité page 58)

[Flum and Grohe, 2006] Flum, J. and Grohe, M. (2006). *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc. (cité page 89)

[Fourdrinoy et al., 2007] Fourdrinoy, O., Grégoire, E., Mazure, B., and Sais, L. (2007). Eliminating redundant clauses in SAT instances. In *International Conference on Integration of AI and OR Techniques in Constraint*

Programming for Combinatorial Optimization Problems (CP-AI-OR), pages 71–83. Springer. (cité page 93)

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. (cité page 40)

[Gavaldà and Thérien, 2009] Gavaldà, R. and Thérien, D. (2009). An algebraic perspective on boolean function learning. In *International Conference on Algorithmic Learning Theory (ALT)*, page 201–215. Springer. (cité page 63)

[Gent and Walsh, 1999] Gent, I. P. and Walsh, T. (1999). Csplib : A benchmark library for constraints. In *Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming*, CP '99, pages 480–481. Springer-Verlag. (cité page 39)

[Glover, 1986] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5) :533–549. (cité page 42)

[Glover and Laguna, 1997] Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers. (cité page 47)

[Gödel, 1931] Gödel, K. (1931). Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für mathematik und physik*, 38(1) :173–198. (cité page 52)

[Goëffon, 2006] Goëffon, A. (2006). *Nouvelles heuristiques de voisinage et mémétiques pour le problème Maximum de Parcimonie*. PhD thesis, LERIA - Université d'Angers. (cité page 65)

[Guo and Niedermeier, 2007] Guo, J. and Niedermeier, R. (2007). Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1) :31–45. (cité page 89)

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research (JMLR)*, 3 :1157–1182. (cité page 66)

[Hajri et al., 2009] Hajri, A., Brin, C., Hunault, G., Lardeux, F., Lemaire, C., Manceau, C., Boureau, T., and Poussier, S. (2009). A «repertoire for repertoire» hypothesis : Repertoires of type three effectors are candidate determinants of host specificity in xanthomonas. *PLoS ONE*, 4 :e6632. (cité pages 23 et 105)

[Hamadi et al., 2011] Hamadi, Y., Monfroy, E., and Saubion, F. (2011). *What is Autonomous Search ?* Springer. (cité page 50)

[Hansen and Jaumard, 1990] Hansen, P. and Jaumard, B. (1990). Algorithms for the maximum satisfiability problem. *Computing*, 44(4) :279–303. (cité page 41)

[Hardy and Ramanujan, 1918] Hardy, G. H. and Ramanujan, S. (1918). Asymptotic formulaæ in combinatory analysis. *Proceedings of the London Mathematical Society*, 2(1) :75–115. (cité page 94)

[Hellerstein and Servedio, 2007] Hellerstein, L. and Servedio, R. A. (2007). On pac learning algorithms for rich boolean function classes. *Theoretical Computer Science*, 384(1) :66–76. (cité page 63)

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press. (cité page 48)

[Hoos and Stützle, 2004] Hoos, H. and Stützle, T. (2004). *Stochastic Local Search : Foundations & Applications*. Morgan Kaufmann Publishers Inc. (cité page 44)

[Hutter et al., 2007] Hutter, F., Hoos, H. H., and Stützle, T. (2007). Automatic algorithm configuration based on local search. In *AAAI*, volume 7, pages 1152–1157. (cité page 50)

[Johnson, 1974] Johnson, D. S. (1974). Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3) :256–278. (cité page 59)

[Karakostas, 2005] Karakostas, G. (2005). A better approximation ratio for the vertex cover problem. In Caires, L., Italiano, G., Monteiro, L., Palamidessi, C., and Yung, M., editors, *Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 1043–1050. Springer Berlin Heidelberg. (cité page 57)

[Karnaugh, 1953] Karnaugh, M. (1953). The map method for synthesis of combinational logic circuits. *American Institute of Electrical Engineers, Part I : Communication and Electronics, Transactions of the*, 72(5) :593–599. (cité page 70)

[Karp, 1972] Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R., Thatcher, J., and Bohlinger, J., editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US. (cité page 55)

[Kratsch and Wahlström, 2010] Kratsch, S. and Wahlström, M. (2010). Preprocessing of min ones problems : A dichotomy. In *Automata, Languages and Programming*, pages 653–665. Springer. (cité pages 41 et 87)

[Land and Doig, 1960] Land, A. H. and Doig, A. G. (1960). An automatic method for solving discrete programming problems. *Econometrica*, 28 :497–520. (cité page 43)

[Lardeux et al., 2006] Lardeux, F., Saubion, F., and Hao, J.-K. (2006). Gasat : a genetic local search algorithm for the satisfiability problem. *Evolutionary Computation*, 14(2) :223–253. (cité page 50)

[Levin, 1973] Levin, L. A. (1973). Universal sequential search problems. *Problems of Information Transmission*, 9(3). (cité page 53)

[Makino et al., 1999] Makino, K., Hatanaka, K.-i., and Ibaraki, T. (1999). Horn extensions of a partially defined boolean function. *SIAM*, 28(6) :2168–2186. (cité page 38)

[Marquis, 2008] Marquis, P. (2008). Knowledge compilation : a sightseeing tour. *Tutorial notes, ECAI*, 8. (cité pages 71 et 112)

[Matsuno et al., 2000] Matsuno, H., Doi, A., Nagasaki, M., and Miyano, S. (2000). Hybrid petri net representation of gene regulatory network. In *Pacific Symposium on Biocomputing*, volume 5, page 87. (cité page 18)

[McCluskey, 1956] McCluskey, E. (1956). Minimization of boolean functions. *Bell System Technical*, 35 :1417–1444. (cité page 70)

[Meinel and Slobodová, 1994] Meinel, C. and Slobodová, A. (1994). On the complexity of constructing optimal ordered binary decision diagrams. In *Mathematical Foundations of Computer Science 1994*, volume 841 of *Lecture Notes in Computer Science*, pages 515–524. Springer. (cité page 37)

[Meyer et al., 2003] Meyer, D., Leisch, F., and Hornik, K. (2003). The support vector machine under test. *Neurocomputing*, 55(1-2) :169–186. (cité page 68)

[Mhedbi, 2010] Mhedbi, N. (2010). *Approches cumulées de phylogénie et d'écologie pour déterminer les bases génétiques de la spécificité d'hôte des bactéries phytopathogènes, cas des Xanthomonas spp.* PhD thesis, PAVE - INH - Université d'Angers. (cité page 20)

[Michener and Sokal, 1957] Michener, C. D. and Sokal, R. R. (1957). A quantitative approach to a problem in classification. *Evolution*, pages 130–162. (cité page 65)

[Nannen and Eiben, 2007] Nannen, V. and Eiben, A. (2007). Efficient relevance estimation and value calibration of evolutionary algorithm parameters. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 103–110. (cité page 50)

[Natarajan, 1987] Natarajan, B. K. (1987). On learning boolean functions. In *Symposium on Theory of Computing (STOC)*, page 296–304. ACM. (cité page 63)

[Nei and Kumar, 2000] Nei, M. and Kumar, S. (2000). *Molecular evolution and phylogenetics*. Oxford University Press, Oxford New York. (cité page 18)

[Obozinski et al., 2006] Obozinski, G., Taskar, B., and Jordan, M. (2006). Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep.* (cité page 70)

[Papadimitriou and Steiglitz, 1982] Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization : Algorithms and Complexity*. Prentice-Hall, Inc. (cité pages 39 et 42)

[Papadimitriou and Yannakakis, 1991] Papadimitriou, C. H. and Yannakakis, M. (1991). Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences (JCSS)*, 43(3) :425–440. (cité page 59)

[Pensa et al., 2006] Pensa, R. G., Robardet, C., and Boulicaut, J.-F. (2006). Supporting bi-cluster interpretation in 0/1 data by means of local patterns. *Journal Intelligent Data Analysis (IDA)*, 10(5) :457–472. (cité pages 63 et 64)

[Quinlan, 1990] Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, 5 :239–266. (cité page 93)

[Rademacher, 1937] Rademacher, H. (1937). On the partition function $p(n)$. *London Mathematical Society*, 2(43) :241–254. (cité page 94)

[Rechenberg, 1965] Rechenberg, I. (1965). Cybernetic solution path of an experimental. *Royal Aircraft Establishment Library Translation*, (1122). (cité page 48)

[Reinelt, 1991] Reinelt, G. (1991). Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4) :376–384. (cité page 39)

[Rymon, 1994] Rymon, R. (1994). On kernel rules and prime implicants. In *AAAI*, volume 94, pages 181–186. (cité page 70)

[Savitch, 1970] Savitch, W. J. (1970). Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences (JCSS)*, 4(2) :177 – 192. (cité page 53)

[Schaad et al., 2005] Schaad, N., Postnikova, E., Lacy, G., Sechler, A., Agarkova, I., Stromberg, P., Stromberg, V., and Vidaver, A. (2005). 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. (cité page 21)

[Sebag and Rouveirol, 1996] Sebag, M. and Rouveirol, C. (1996). Constraint inductive logic programming. (cité page 112)

[Simon, 2003] Simon, R. (2003). *Design and Analysis of DNA Microarray Investigations*. Statistics for Biology and Health. Springer. (cité page 18)

[Srinivas and Patnaik, 1994] Srinivas, M. and Patnaik, L. M. (1994). Genetic algorithms : A survey. *Computer*, 27(6) :17–26. (cité page 49)

[Stockmeyer, 1976] Stockmeyer, L. J. (1976). The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1) :1–22. (cité page 56)

[Stockmeyer and Meyer, 1973] Stockmeyer, L. J. and Meyer, A. R. (1973). Word problems requiring exponential time : Preliminary report. In *Symposium on Theory of Computing (STOC)*, pages 1–9. (cité page 56)

[Tibshirani, 1994] Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58 :267–288. (cité page 68)

[Tou and González, 1974] Tou, J. and González, R. (1974). *Pattern recognition principles*. Applied mathematics and computation. Addison-Wesley Pub. Co. (cité page 64)

[Turing, 1936] Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58 :345–363. (cité page 51)

[Umans, 2001] Umans, C. (2001). The minimum equivalent DNF problem and shortest implicants. *Journal of Computer and System Sciences (JCSS)*, 63(4) :597–611. (cité page 70)

[Valiant, 1984] Valiant, L. G. (1984). A theory of the learnable. *Magazine Communications of the ACM*, 27(11) :1134–1142. (cité page 63)

[Vauterin et al., 1995] Vauterin, L., Hoste, B., Kersters, K., and Swings, J. (1995). Reclassification of xanthomonas. *International Journal of Systematic Bacteriology (IJSEM)*, 45 :472–489. (cité page 21)

[Zou and Hastie, 2005] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 67(2) :301–320. (cité page 68)

Thèse de Doctorat

Fabien-Sothéa CHHEL

Problème de caractérisation multiple

Application à la détection de souches bactériennes phytopathogènes

Multiple charaterization problem

Application to detection of phytopatogenous bacterial strains

Résumé

L'informatique joue un rôle de plus en plus prépondérant dans l'analyse et la compréhension des données biologiques. Dans cette thèse, nous présentons le problème de caractérisation multiple qui aborde, sous l'angle de l'optimisation combinatoire, la détection de souches bactériennes phytopathogènes. En effet, certaines bactéries sont responsables de pathologies sur une large gamme de cultures diminuant ainsi la valeur marchande des semences. Des tests de diagnostic peuvent être alors conçus et nécessitent l'identification de caractères observables discriminants. Nous avons développé plusieurs méthodes de résolution exactes (*branch and bound* et programmation linéaire) et approchées (métaheuristiques) afin de minimiser le nombre de caractères à traiter. Précisément, en considérant un sous-ensembles d'individus représentatifs de différents groupes, il s'agit d'établir pour chacun de ces groupes une caractérisation (sous forme de formule propositionnelle) permettant de distinguer les individus du groupe par rapport à ceux des autres groupes. Nous utilisons le terme « caractérisation multiple » pour rendre compte de l'aspect mutuel de ces discriminations au sens « un groupe versus le reste des groupes » et ce pour chaque groupe. Nos travaux ont été validés expérimentalement sur des jeux de données et une étude approfondie de la complexité de ce problème a été menée.

Abstract

Computer science plays an increasingly important role in the analysis and understanding of biological data. In this thesis, we present the multiple characterization problem that addresses, as a combinatorial optimization problem, the detection of plant pathogenic bacterial strains. Indeed, some bacteria are responsible for diseases on a wide range of crops and decrease the market value of the seeds. Diagnostic tests can be designed and require the identification of discriminant observable characters. We have developed several exact methods (branch and bound, linear programming) and approximate (metaheuristics) to minimize the number of characters to deal with. Specifically, considering a subset of individuals representing different groups, the aim for each group is to find a characterization (as a propositional formula) to distinguish the individuals of the group compared to those of the other groups. We use the term "multiple characterization" to highlight the mutual aspect of theses discriminations in the sense "versus a group the other groups" and that for each group. Our work has been validated experimentally on data sets and an in-depth study of the problem's complexity has been conducted.

Mots clés

optimisation combinatoire, caractérisation de données.

Keywords

combinatorial optimization, data charaterization.

L'UNIVERSITÉ NANTES ANGERS LE MANS