UNIVERSITÉ D'ARTOIS

Fouille de données par contraintes

THÈSE

présentée et soutenue publiquement le 13 septembre 2018

en vue de l'obtention du

Doctorat de l'Université d'Artois

(Spécialité Informatique)

par

Abdelhamid Boudane

Composition du jury

Frédéric Saubion	Professeur à l'université d'Angers (Rapporteur)
Thi-Bich-Hanh Dao	Maître de conférence (HDR) à l'université d'Orléans (Rapporteur)
Salima Benbernou	Professeur à l'université de Paris Descartes (Examinateur)
Lakhdar Sais	Professeur à l'université d'Artois (Directeur de thèse)
Said Jabbour	Maître de conférence à l'université d'Artois (Encadrant)
Yakoub Salhi	Maître de conférence à l'université d'Artois (Encadrant)

Centre de Recherche en Informatique de Lens – CNRS UMR 8188
Université d'Artois, rue Jean Souvraz, S.P. 18 F-62307, Lens Cedex France
Secrétariat : Tél.: +33 (0)3 21 79 17 23 – Fax : +33 (0)3 21 79 17 70
http://www.cril.univ-artois.fr

Mise en page avec memcril (B. Mazure, CRIL) et thloria (D. Roegel, LORIA).

Remerciements

J'adresse mes premiers remerciements aux rapporteurs, madame Thi-Bich-Hanh Dao et monsieur Frédéric Saubion qui m'ont donné beaucoup de conseils qui m'ont permis d'améliorer ce rapport de thèse.

Je remercie également le professeur Salima Benbernou pour m'avoir fait l'honneur d'examiner ce travail.

Je remercie infiniment mon directeur de thèse monsieur Lakhdar Saïs et mes encadrants Said Jabbour et Yakoub Salhi pour leur encadrement exemplaire et pour leurs conseils et remarques qui m'ont permis de réaliser ce travail.

Merci à tous les membres du CRIL pour leurs disponibilités et pour la bonne ambiance au sein du laboratoire ainsi qu'à tous les doctorants.

Je remercie également l'EMP pour son soutien financier, très important pour mener à bien ce travail.

Un grand Merci à toute ma famille et spécialement mes parents, ils étaient toujours présent pour m'encourager.

Un grand merci à mes amis Yacine, Yazid, Nizar, Khalil avec qui j'ai passé des moments inoubliables.

Enfin, merci à tous les amis d'Arras.

Je dédie cette thèse
à ma famille.

Table des matières

Introduction générale

Le domaine de la fouille de données (ou découverte de connaissances à partir des données) a connu des développements importants ces vingt dernières années. Cette évolution a été guidée particulièrement par des applications importantes dans divers secteurs industriels qui vont de la finance, au commerce en passant par les réseaux sociaux et la bio-informatique. Nous pouvons distinguer plusieurs approches qui visent l'extraction des informations cachées dans de grandes bases de données comme la fouille de règles d'association ou la classification non supervisée « clustering ».

Le clustering est un problème fondamental dans le domaine de la fouille de données. Il consiste à partitionner un ensemble d'objets dans des catégories ou des groupes de telle sorte que les objets qui appartiennent à un même groupe ont des propriétés similaires par rapport à des critères bien définis, et que les groupes soient les plus dissemblables possibles. Le problème de clustering a été largement étudié vu son importance dans divers domaines comme en biologie [WEL+10], dans l'analyse des images [MK95], et dans le commerce [WXL99], etc. Par conséquent, plusieurs algorithmes ont été proposés, citant comme exemples l'algorithme k-means [Mac67], les algorithmes hiérarchiques [WJ63] et les algorithmes basées sur la densité [EKSX96].

La fouille de règles d'association est une autre tâche très importante en fouille de données. Elle consiste à extraire des relations intéressantes cachées dans des bases de données transactionnelles. Ces relations sont présentées sous forme d'implications $X \rightarrow Y$, appelées règles d'association, où X et Y sont des ensembles d'items. La première application sur des bases de transactions effectuées par des clients dans un magasin [AS94], a visé la recherche des relations entre les produits permettant l'analyse des différents comportements des clients. Par exemple, une règle stipulant que 80% des clients qui achètent du pain achètent aussi du lait, permet d'identifier un lien fort entre le lait et le pain. Ce type de relations permet, entre autres, à un manager de magasin de bien organiser les rayons. Depuis cette application, plusieurs nouveaux domaines d'application ont été identifiés comme la bio-informatique [AGF+09], le diagnostic médical [KW14], la détection d'intrusions [MD12], la fouille du web [Kaz09a] et l'analyse des données scientifiques [LS16]. Ce large spectre d'applications a poussé les recherches dans ce domaine. On peut distinguer deux problématiques majeures dans ce domaine de recherche. La première est relative à l'extraction des règles d'association les plus intéressantes, quant à la seconde, elle concerne la définition de nouveaux types de règles d'association ou de relations qui ne peuvent pas être déduites à partir de l'ensemble de règles d'association standard. En effet, plusieurs contributions ont été proposées pour, d'un côté, trouver un sous-ensemble de règles intéressantes [BPT+00, Zak04] et d'un autre côté, définir d'autres types de règles comme les règles d'exception ou les règles inattendues [Suz02, PT99]. Pour chaque nouvelle définition de règles, de nouveaux algorithmes dédiés ont été proposés.

Les données considérées en fouille de données peuvent être de différents types, à savoir, les données sous forme de transactions, séquences, arbres, graphes, vecteurs, documents, etc. Des algorithmes de clustering spécifiques à chaque type de données ont été proposés dans la littérature en exploitant les caractéristiques et les propriétés de chaque type. Aujourd'hui, nous vivons dans une aire de don-

nées issues de divers domaines qui vont du web à la santé en passant par le commerce et l'astronomie. De plus, l'évolution des technologies et en particulier les moyens de calcul et de stockage a permis de collecter des volumes importants de données de plus en plus complexes et hétérogènes. La plupart des algorithmes de clustering manipulent des données exprimées sous forme de vecteurs numériques ou catégoriels. Or, il existe de nombreuses applications où les données sont plutôt symboliques et complexes. À titre d'exemples, on peut citer les données issues des préférences d'agents sur des modèles de voitures qu'ils souhaitent acquérir, ou encore les données qui peuvent être collectées suite à des sondages ou questionnaires. Pour ce type de données, la logique formelle avec sa puissance d'expression peut s'avérer être un moyen adéquat pour une représentation plus compacte et facilement manipulable via des outils puissants développés pour résoudre différents problèmes en intelligence artificielle. En considérant cette représentation logique, le principal défi concerne l'adaptation des algorithmes de clustering existants pour manipuler des données exprimées par des formules logiques. Dans ce cadre, notre première contribution consiste à répondre à cette question.

Dans [RNOH11], les auteurs ont souligné que les contraintes font souvent partie de la spécification de plusieurs problèmes de fouille de données. Cette observation a conduit à un nouveau domaine de recherche actif et multidisciplinaire initié dans [RGN08], et permettant une fertilisation croisée entre la fouille de données et l'intelligence artificielle. Deux modèles de représentation et de résolution d'IA ont été utilisés pour modéliser et résoudre plusieurs problèmes de fouille de données, à savoir la programmation par contrainte (CP) et la satisfiabilité propositionnelle (SAT). Parmi ces problèmes, nous pouvons citer l'extraction de motifs [RGN08, JSS15a] et le clustering [DRS10a, DDV13]. Cette nouvelle approche offre un modèle de représentation flexible et déclaratif. De nouvelles contraintes nécessitent souvent de nouvelles implémentations pour la plupart des problèmes de fouille de données, alors que ces mêmes contraintes peuvent être facilement intégrées dans de tels modèles déclaratifs. Suivant cette tendance de recherche, notre deuxième défi est de proposer un cadre basé sur la satisfiabilité propositionnelle pour modéliser et résoudre efficacement les problèmes de fouille de différentes variantes de règles d'association en une seule étape. Les différentes tâches sont encodées comme une formule propositionnelle dont les modèles correspondent aux règles à fouiller.

Cette thèse se décompose en deux grandes parties. La première est réservée à la présentation de l'état de l'art des différentes notions et algorithmes liés à nos contributions, qui concernent principalement la satisfiabilité propositionnelle, le clustering et la fouille de règles d'association. Dans la deuxième partie, nous présentons nos contributions concernant le clustering de formules propositionnelles et la fouille de différentes variantes de règles d'association en utilisant la satisfiabilité propositionnelle.

Ce travail contient cinq chapitres. Dans le premier, nous donnons une description des principales notions liées au problème SAT nécessaires pour comprendre nos différentes contributions. En effet, nous commençons par un rappel de la syntaxe et de la sémantique de la logique propositionnelle. Puis, nous présentons le problème de la satisfiabilité propositionnelle. Enfin, nous décrivons quelques algorithmes proposés pour résoudre ce problème. Le deuxième chapitre est réservé à la description des différentes méthodes proposées pour le clustering. Nous présentons d'abord le clustering. Puis, nous montrons comment les mesures de distances changent en changeant le type de données. Ensuite, nous présentons les plus connus des algorithmes proposés pour le clustering en montrant les avantages et les inconvénients de chacun. Enfin, nous donnons un aperçu de quelques approches de clustering sous contraintes. Dans le troisième chapitre, les différentes notions liées à la fouille de règles d'association sont présentées. En effet, nous commençons par présenter le problème d'extraction de motifs ensemblistes. Puis, nous montrons comment ces motifs sont utilisés pour extraire les règles d'association. Ensuite, nous présentons les différents travaux qui traitent la redondance de règles d'association en décrivant les différentes variantes de règles qui ont été définies. Enfin, nous décrivons les modélisations basées sur la programmation par

contraintes et la satisfiabilité propositionnelle du problème de fouille de motifs ensemblistes. Dans le quatrième chapitre, nous présentons nos contributions au clustering de formules propositionnelles. Nous commençons d'abord par un exemple réel de motivation où nous montrons l'utilité de la représentation logique. Ensuite, nous décrivons les différentes contributions. Nous présentons d'abord une adaptation de deux algorithmes de clustering, à savoir, le k-means et l'algorithme hiérarchique ascendant. Ensuite, nous montrons les inconvénients de ces adaptations et nous proposons un algorithme hiérarchique descendant pour le clustering des formules propositionnelles qui garantit une solution si elle existe. Enfin, une étude expérimentale est effectuée pour montrer le comportement des différentes solutions proposées. Dans le dernier chapitre, nous proposons un cadre déclaratif basé sur la satisfiabilité propositionnelle pour la fouille des différentes variantes de règles d'association. Nous montrons comment modéliser les différentes contraintes liées à chaque variante. Nous proposons ainsi le premier encodage qui permet l'extraction des motifs minimaux et nous montrons comment combiner cet encodage avec d'autres encodages de contraintes pour extraire les règles non redondantes. Nous montrons aussi comment adapter les encodages SAT existants de la contrainte de cardinalité pour encoder la contrainte de cardinalité conditionnelle et nous utilisons cet encodage pour accélérer l'extraction des règles non redondantes. Enfin, une large étude expérimentale est effectuée pour montrer le comportement de notre approche par rapport aux approches dédiées et nous montrons que notre approche est plus performante dans l'extraction des différentes variantes de règles d'association. Nous terminons ce mémoire par une conclusion générale et de nombreuses perspectives.

Ce travail a donné lieu aux publications suivantes :

1. SAT-Based Data Mining. International Journal on Artificial Intelligence Tools 27(1) : 1-24 (2018).
2. Clustering Complex Data Represented as Propositional Formulas. PAKDD (2) 2017 : 441-452.
3. Enumerating Non-redundant Association Rules Using Satisfiability. PAKDD (1) 2017 : 824-836.
4. Une approche logique pour la fouille de règles d'association. EGC 2017 : 357-362.
5. Une Approche Basée Sur SAT Pour l'Énumération Des Règles d'Association Non Redondantes. JFPC 2017.
6. A SAT-Based Approach for Mining Association Rules. IJCAI 2016 : 2472-2478

Première partie

État de l’art

Chapitre 1

Satisfiabilité propositionnelle

Sommaire

La satisfiablité propositionnelle (SAT) est un problème de décision relatif à la logique propositionnelle classique dont les solutions ont été utilisées dans plusieurs applications industrielles pour résoudre des problèmes très complexes durant ces deux dernières décennies. L'approche fondée sur la modélisation en SAT fournit un langage générique qui peut être utilisé pour exprimer des problèmes complexes (NP-complets) dans plusieurs domaines, comme la vérification formelle [VB03], la planification [Mar09], la configuration [MBC09] et la fouille de données[MBC+12]. Cette approche utilise des solutions performantes pour SAT, communément appelées solveurs, pour la recherche automatique des solutions du problème. Cela permet d'éviter le redéveloppement de nouvelles solutions algorithmiques pour chacune des applications tout en garantissant de hautes performances.

Le problème de satisfiabilité propositionnelle, consiste à décider s'il existe une assignation des variables propositionnelles qui rend une formule donnée de la logique propositionnelle vraie. En théorie de la complexité algorithmique, SAT est le problème NP-complet de référence [Coo71], en conséquence, la résolustion efficace de SAT est devenue un axe de recherche très important. Il est important de noter que de nombreux algorithmes ont été proposés durant les deux dernières décennies pour améliorer la résolution de SAT [Een05, AS09].

Dans ce chapitre nous présentons les fondements de la satisfiabilité propositionnelle ainsi que des algorithmes de résolution dédiés au problème SAT. Pour mieux comprendre SAT nous commençons d'abord par la présentation de la logique propositionnelle.

1.1 Logique propositionnelle

Dans cette section, nous décrivons la syntaxe et la sémantique de la logique propositionnelle.

1.1.1 Syntaxe

Dans la syntaxe, il s'agit de décrire la structure des formules propositionnelles bien formées en définissant le langage permettant de les construire.

Une variable propositionnelle est une variable ayant comme domaine $\{0, 1\}$. Le chiffre 0 est utilisé pour représenter le $faux$ et 1 pour représenter le $vrai$. Soit Prop un ensemble de variables propositionnelles. Nous utilisons les lettres minuscules p, q, r, etc pour noter les éléments de Prop.

Définition 1.1 (Formule propositionnelle)**.** *L'ensemble des formules propositionnelles, noté* Form*, est défini inductivement en commençant par* Prop*, la constante $\bot$, la constante $\top$ et en utilisant les connecteurs logiques $\neg$ (non), $\wedge$ (et), $\vee$ (ou), $\rightarrow$ (implication) et $\leftrightarrow$ (équivalence) de telle sorte que :*

1. *les variables propositionnelles, $\bot$ et $\top$ sont des formules ;*
2. *si $\mathcal{F}$ est une formule alors $\neg\mathcal{F}$ l'est aussi ;*
3. *si $\mathcal{F}$ et $\mathcal{F}'$ sont des formules alors les expressions suivantes sont des formules :*
   - *(a)* $(\mathcal{F} \wedge \mathcal{F}')$;
   - *(b)* $(\mathcal{F} \vee \mathcal{F}')$;
   - *(c)* $(\mathcal{F} \rightarrow \mathcal{F}')$;
   - *(d)* $(\mathcal{F} \leftrightarrow \mathcal{F}')$.

Notons que nous pouvons réstreindre le langage aux deux connecteurs logiques $\neg$ et $\wedge$ car nous avons les équivalances suivantes :

— $A \vee B \equiv \neg(\neg A \wedge \neg B)$;
— $A \rightarrow B \equiv \neg A \vee B$;
— $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$.

Le symbole $\equiv$ est utilisé pour noter l'équivalence logique.

Exemple 1.1. *Soient $\mathcal{F}_1$ et $\mathcal{F}_2$ deux expressions construites sur l'ensemble de variables propositionnelles $\{p, q, r\}$ tel que $\mathcal{F}_1 = (p \wedge q)(\neg r)$ et $\mathcal{F}_2 = (p \wedge q) \vee (\neg r)$. Alors $\mathcal{F}_1$ n'appartient pas à* Form*, contrairement à $\mathcal{F}_2$.*

Définition 1.2 (Littéral)**.** *Un littéral est une variable propositionnelle p (littéral positif) ou sa négation $\neg p$ (littéral négatif), noté aussi $\overline{p}$. Les deux littérraux p et $\overline{p}$ sont dits complémentaires.*

Étant donné une formule $\mathcal{F}$, nous utilisons $\mathcal{P}(\mathcal{F})$ (resp. $\mathcal{L}(\mathcal{F})$) pour noter l'ensemble des variables propositionnelles (resp. les littéraux) qui apparaissent dans $\mathcal{F}$.

1.1.2 Sémantique

La sémantique d'une logique formelle permet l'étude du sens des formules bien formées.

Définition 1.3 (Interprétation Booléenne)**.** *Une interprétation Booléenne $\mathcal{I}$ d'une formule propositionnelle $\mathcal{F}$ est une fonction de $\mathcal{P}(\mathcal{F})$ dans $\{0, 1\}$. L'interprétation $\mathcal{I}(\mathcal{F})$ est définie par la valeur de vérité donnée à chacune des variables de $\mathcal{F}$. Elle est étendue inductivement aux formules en utilisant les règles suivantes :*

1. $\mathcal{I}(\bot) = 0$;
2. $\mathcal{I}(\top) = 1$;
3. $\mathcal{I}(\neg\mathcal{F}) = 1 - \mathcal{I}(\mathcal{F})$;
4. $\mathcal{I}(\mathcal{F} \wedge \mathcal{G}) = min(\mathcal{I}(\mathcal{F}), \mathcal{I}(\mathcal{G}))$;
5. $\mathcal{I}(\mathcal{F} \vee \mathcal{G}) = max(\mathcal{I}(\mathcal{F}), \mathcal{I}(\mathcal{G}))$;
6. $\mathcal{I}(\mathcal{F} \rightarrow \mathcal{G}) = max(1 - \mathcal{I}(\mathcal{F}), \mathcal{I}(\mathcal{G}))$;

l'interprétation $\mathcal{I}$ est souvent représentée comme un ensemble de littéraux p si $\mathcal{I}(p) = 1$ et $\neg p$ si $\mathcal{I}(p) = 0$.
Soit $\mathcal{S}(\mathcal{F})$ l'ensemble de toutes les interprétations possibles d'une formule $\mathcal{F}$. Si $|\mathcal{P}(\mathcal{F})| = n$ alors $|\mathcal{S}(\mathcal{F})| = 2^n$.

Exemple 1.2. *Soit $\mathcal{F}$ une formule construite sur deux variables avec $\mathcal{P}(\mathcal{F}) = \{p, q\}$. Nous distingons quatre interprétations totales possibles : $\mathcal{I}_1 = \{p, q\}$, $\mathcal{I}_2 = \{p, \neg q\}$, $\mathcal{I}_3 = \{\neg p, q\}$ et $\mathcal{I}_4 = \{\neg p, \neg q\}$.*

Définition 1.4 (Modèle, contre-modèle)**.** *Un modèle d'une formule $\mathcal{F}$ est une interprétation Booléenne $\mathcal{I}$ qui satisfait $\mathcal{F}$, c-à-d, $\mathcal{I}(\mathcal{F}) = 1$ et on note $\mathcal{I} \models \mathcal{F}$. Inversement, si $\mathcal{I}$ falsifie $\mathcal{F}$, c-à-d, $\mathcal{I}(\mathcal{F}) = 0$ alors $\mathcal{I}$ est dite contre-modèle de $\mathcal{F}$ et on note $\mathcal{I} \not\models \mathcal{F}$.*

Exemple 1.3. *Soit la formule $\mathcal{F} = (p \wedge \neg q) \vee r$ ainsi que les deux interprétations associées $\mathcal{I}_1 = \{p, q, \neg r\}$ et $\mathcal{I}_2 = \{p, \neg q, \neg r\}$. Nous avons $\mathcal{I}_1(\mathcal{F}) = max(min(1, 0), 0) = max(0, 0) = 0$ et $\mathcal{I}_2(\mathcal{F}) =$ max(min(1, 1), 0) = $max(1, 0) = 1$. Par conséquent, l'interprétation $\mathcal{I}_2$ est un modèle de $\mathcal{F}$ alors que $\mathcal{I}_1$ est un contre-modèle de cette dernière.*

Nous utilisons $\mathcal{M}(\mathcal{F})$ pour noter l'ensemble des modèles d'une formule $\mathcal{F}$.
Une formule propositionnelle $\mathcal{F}$ est une tautologie (valide ou théorème) si toutes les interprétations de $\mathcal{F}$ sont des modèles de $\mathcal{F}$, c-à-d, $\mathcal{S}(\mathcal{F}) = \mathcal{M}(\mathcal{F})$.

Définition 1.5 (Formule satisfiable, insatisfiable)**.** *Une formule $\mathcal{F}$ est satisfiable si elle admet au moins un modèle, c-à-d, $\mathcal{M}(\mathcal{F}) \neq \emptyset$. En revanche, $\mathcal{F}$ est dite insatisfiable si elle n'admet aucun modèle, c-à-d, $\mathcal{M}(\mathcal{F}) = \emptyset$.*

Exemple 1.4. *Les formules $\mathcal{F}_1 = (p \wedge \neg q) \vee r$, $\mathcal{F}_2 = (\neg p \vee q) \wedge (\neg q \wedge p)$ et $\mathcal{F}_3 = (p \rightarrow p)$ sont respectivement satisfiable ($\mathcal{I} = \{p, \neg q, r\}$ est un modèle), insatisfiable et valide.*

Définition 1.6 (Conséquence logique)**.** *Une formule propositionnelle $\mathcal{G}$ est une conséquence logique d'une formule $\mathcal{F}$, noté $\mathcal{F} \models \mathcal{G}$, ssi $\mathcal{M}(\mathcal{F}) \subseteq \mathcal{M}(\mathcal{G})$. Si $\mathcal{F} \models \mathcal{G}$ et $\mathcal{G} \models \mathcal{F}$ alors $\mathcal{G}$ et $\mathcal{F}$ sont dites équivalentes et on note $\mathcal{G} \equiv \mathcal{F}$.*

Nous décrivons maintenant un théorème montrant que prouver la conséquence logique revient à prouver l'insatisfiabilité d'une formule.

Théorème 1.1. *Soit $\mathcal{F}$ et $\mathcal{G}$ deux formules propositionnelles. $\mathcal{F} \models \mathcal{G}$ ssi la formule $\mathcal{F} \wedge \neg\mathcal{G}$ est insatisfiable.*

1.1.3 Formes normales

Les formes normales ont été définies afin de simplifier le language propositionnelle et de pouvoir utiliser un ensemble de règles sur les formules propositionnelles. Les trois formes les plus utilisées sont la forme normale négative (NNF), la forme normale conjonctive (CNF) et la forme normale disjonctive (DNF).

Définition 1.7 (Forme NNF)**.** *Une formule est sous forme normale négative (NNF) si elle est exprimée exclusivement avec $\wedge$, $\vee$ et $\neg$.*

Définition 1.8 (Forme DNF)**.** *Une formule DNF est une disjonction de monômes où un monôme est une conjonction de littéraux.*

$$\mathcal{F} = \bigvee_{m \in \mathcal{F}} (\bigwedge_{l_i \in m} l_i) \tag{1.1}$$

Il est clair qu'un monôme est insatisfiable ssi au moins un de ses littéraux est évalué à 0, vu l'interprétation de la conjonction. Une formule DNF est satisfiable s'il existe au moins une interprétation qui rend au moins un de ses monômes satisfiable.

Définition 1.9 (Forme CNF)**.** *Une formule sous forme normale conjonctive (CNF) est une conjonction de clauses où une clause est une disjonction de littéraux.*

$$\mathcal{F} = \bigwedge_{c \in \mathcal{F}} (\bigvee_{l_i \in c} l_i) \tag{1.2}$$

Une formule CNF peut être considérée comme un ensemble de clauses et chaque clause comme un ensemble de littéraux. La taille d'une formule CNF $\mathcal{F}$ correspond à $\sum_{c \in \mathcal{F}} |c|$ où $|c|$ est le nombre de littéraux dans c.

Une formule CNF $\mathcal{F}$ est satisfaite par une interprétation $\mathcal{I}$ ssi chaque clause de $\mathcal{F}$ est satisfaite par au moins un littéral de $\mathcal{I}$: $\forall c \in \mathcal{F}, \exists l \in \mathcal{I}$ tel que $l \in c$.

Exemple 1.5. *La formule $\mathcal{F}_1 = (p \vee \neg q) \wedge (\neg r \vee p) \wedge (\neg r \vee \neg q)$ est sous forme normale conjonctive (CNF) tandis que la formule $\mathcal{F}_2 = (p \wedge \neg q) \vee (\neg r \wedge p) \vee (\neg r \wedge \neg q)$ est sous forme normale disjonctive (DNF).*

On peut distinguer les différents types de clauses suivants :
- **Clause unitaire** : une clause c est dite unitaire (mono-littéral) ssi elle contient seulement un seul littéral.
- **Clause binaire** : une clause c est dite binaire ssi elle contient exactement deux littéraux.
- **Clause de Krom** : une clause de Krom est une clause qui contient au plus deux littéraux.
- **Clause positive (resp. négative)** : une clause positive (resp. clause négative) est une clause qui ne contient pas de littéraux négatifs (resp. positifs). Une clause constituée de littéraux positifs et négatifs est appelée clause mixte.
- **Clause vide** : une clause c est vide, notée $\perp$, est une clause ne contenant aucun littéral. Elle est par définition insatisfiable.
- **Clause tautologique** : une clause tautologique est une clause contenant un littéral et son complémentaire. Elle est par définition vraie.
- **Clause de Horn (resp. reverse-Horn)** : une clause de Horn (resp. reverse-Horn) est une clause qui contient au plus un littéral positif (resp. négatif).

Exemple 1.6. *Les clauses $\neg p$, $p \vee \neg r$, $p \vee q$, $\neg q \vee \neg r$, $p \vee \neg p$, $\neg p \vee q \vee \neg r$ et $p \vee \neg q \vee r$ sont respectivement unitaire, binaire, positive, négative, tautologique, Horn et reverse-Horn.*

Notons que toute formule propositionnelle $\mathcal{F}$ peut être transformée en une formule équivalente $\mathcal{G}$ sous une des formes normales NNF, DNF et CNF. Cette transformation est effectuée via certaines règles comme les lois de distributivité et les lois de Morgan, etc.

Nous considérons d'abord que toutes les implications et les équivalences sont réécrites en utilisant les connecteurs logique $\wedge$, $\vee$ et $\neg$. Maintenant, en applicant les règles suivantes, d'une façon non déterministe jusqu'à ce qu'aucune règle ne peut être appliquée, nous pouvons transformer toute formule propositionnelle en une formule équivalente sous forme normale négative. Cette transformation est assez simple et peut être effectuée en un temps linéaire :

1. $\bot \wedge \mathcal{F} \equiv \bot$; $\bot \vee \mathcal{F} \equiv \mathcal{F}$; $\neg\bot \equiv \top$
2. $\top \wedge \mathcal{F} \equiv \mathcal{F}$; $\top \vee \mathcal{F} \equiv \top$; $\neg\top \equiv \bot$
3. $\neg(\mathcal{F} \wedge \mathcal{G}) \equiv \neg\mathcal{F} \vee \neg\mathcal{G}$; $\neg(\mathcal{F} \vee \mathcal{G}) \equiv \neg\mathcal{F} \wedge \neg\mathcal{G}$; $\neg\neg\mathcal{F} \equiv \mathcal{F}$

Les règles dans (1 et 2) permettent l'évaluation des deux constantes $vrai$ et $faux$ tandis que les règles dans (3) nous permettent de pousser la négation vers l'intérieur jusqu'aux variables propositionnelles.

La plupart des méthodes de résolutions du problème de satisfiabilité propositionnelle acceptent et utilisent une formule sous forme normale conjonctive. Toute formule NNF peut être transformée en une formule CNF equivalente. Cependant, cette translation n'est pas efficace car elle génère, dans le pire cas, une formule CNF de taille exponentielle. Étant donné une formule NNF, l'algorithme standard qui permet cette transformation applique simplement la règle de réécriture suivante d'une manière non déterministe jusqu'à ce qu'elle ne puisse plus être appliquée.

$$(A_1 \wedge ... \wedge A_n) \vee B_1 \vee ... \vee B_m \equiv \bigwedge_{i=1}^{n} (A_i \vee B_1 \vee ... \vee B_m)$$

Cette transformation n'est pas très utile en pratique à cause du nombre exponentielle de clauses qu'elle peut génèrée. En utilisant l'approche proposée initialement par Tseitin [Tse68], nous pouvons transformer toute formule propositionnelle $\mathcal{F}$ en une formule CNF $\mathcal{F}_c$ équivalente pour la satisfiabilité, c-à-d, $\mathcal{F}$ est satisfiable si et seulement si $\mathcal{F}_c$ est satisfiable. Cette transformation est linéaire en temps et en espace. Elle consiste à ajouter des variables supplémentaires pour représenter des sous-formules. Cet ajout de variables permet d'éviter une explosion combinatoire résultant de la distributivité du $\wedge$ par rapport au $\vee$ et vice versa. La transformation procède en deux étapes. Tout d'abord, la formule $\mathcal{F}$ est transformée en une formule NNF $\mathcal{F}_n$. Puis, des variables supplémentaires sont ajoutées pour représenter les sous-formules de la forme $l_1 \wedge ... \wedge l_k$ ou $l_1 \vee ... \vee l_k$, induisant des définitions de la forme $p \leftrightarrow (l_1 \wedge ... \wedge l_k)$ ou $p \leftrightarrow (l_1 \vee ... \vee l_k)$ où p est la nouvelle variable. La formule CNF résultante $\mathcal{F}_c$ est obtenue en remplaçant les sous-formules par les variables correspondantes dans $\mathcal{F}_n$ et en ajoutant les clauses qui représentent les définitions introduites.
En considérant la polarité [PG86], nous pouvons simplifier la transformation précédente en remplaçant les équivalences par des implications dans les sous-formules comme suit : $p \rightarrow (q_1 \wedge ... \wedge q_k)$; $p \rightarrow (q_1 \vee ... \vee q_k)$ tout en gardant l'équisatisfiabilité et en réduisant le nombre de clauses dans $\mathcal{F}_c$. Il est à noter que chaque implication est réécrite sous forme d'une clause.

Exemple 1.7. *Soit la formule propositionnelle $\mathcal{F} = p \rightarrow \neg(q \vee r) \vee f$. Une formule NNF équivalente à $\mathcal{F}$ est $\mathcal{F}_n = \neg p \vee (\neg q \wedge \neg r) \vee f$. Une formule CNF équivalente à $\mathcal{F}$ est $\mathcal{F}_e = (\neg q \vee \neg p \vee f) \wedge (\neg r \vee \neg p \vee f)$. En utilisant l'encodage de Tseitin sur la formule $\mathcal{F}_n$, nous rajoutons tout d'abord la formule $p_1 \leftrightarrow (\neg q \wedge \neg r)$ où p_1 est une variable supplémentaire. Enfin, une formule CNF $\mathcal{F}_c$ équisatisfiable à $\mathcal{F}$ est $\mathcal{F}_c = (p_1 \vee q \vee r) \wedge (\neg p_1 \vee \neg q) \wedge (\neg p_1 \vee \neg r) \wedge (\neg p \vee p_1 \vee f)$.*

1.2 Problème SAT

SAT est un problème de décision très important en théorie de la complexité. Il est devenu le problème NP-complet de référence après la preuve d'appartenance à cette classe de complexité donnée par Cook en 1971 [Coo71]. Plusieurs problèmes complexes dans de nombreux domaines ont pu être encodés où réduits vers SAT. Ceci a poussé la recherche de solutions algorithmiques éfficaces pour SAT. Dans cette section nous définissons le problème SAT et nous présentons quelques algorithmes qui permettent sa résolution.

Définition 1.10 (Problème SAT)**.** *Le problème SAT est un problème de décision qui consiste à déterminer si une formule propositionnelle sous forme normale conjonctive (CNF) $\mathcal{F}$ admet un modèle.*

Exemple 1.8. *Soit $\mathcal{F}$ la formule CNF suivante :*

$$(\neg p \vee \neg q \vee r) \wedge (p \vee \neg q) \wedge (q \vee r) \wedge (\neg r \vee \neg p)$$

Question : $\mathcal{F}$ admet elle au moins un modèle ?
Réponse : Oui car l'interprétation $\mathcal{I} = \{\neg p, \neg q, r\}$ est un modèle de $\mathcal{F}$.

1.2.1 Classes polynomiales de SAT

Une classe polynomiale de SAT est un type de formules pour lesquelles le problème SAT peut être résolu en temps polynomial. Reconnaître cette catégorie de formules a d'autant d'intérêts théoriques que partiques surtout quand il s'agit de formules qui encodent des problèmes réels. De plus, on peut exploiter ces classes pour mieux résoudre le cas général [GU89, Bac02]. Nous présontons dans cette section les plus connues des classes polynomiales qui ont été définies dans la littérature.

Définition 1.11 ($k - SAT$)**.** *Le problème $k - SAT$ consiste à déterminer si une formule CNF composée uniquement de clauses de taille k admet un modèle.*

Décider la satisfiabilité d'une formule $k - SAT$ est polynomial pour $k = 2$ (clauses binaire) et $NP - Complet$ pour $k \geqslant 3$. Notons que toute formule CNF peut se réduire à une formule $3 - SAT$ équivalente pour la satisfiabilité en rajoutant de nouvelles variables de la même manière que la transformation d'une formule vers la forme CNF (voir section 1.1.3). On retrouve souvent les clauses binaires dans la plupart des instances venant des applications réelles. Un traitement particulier sur ce type de clauses améliore considérablement l'efficacité des algorithmes de résolution.

Les formules de Horn (resp. reverse Horn) sont des formules avec des clauses qui contiennent au plus un littéral positive (resp. au plus un littéral négative). Ces formules forment aussi une classe polynomiale. Notons que les clauses de Horn sont la base de la programmation logique et on peut citer comme exemple la version initiale du langage $PROLOG$ qui n'acceptait que des règles de la forme $p \wedge q \rightarrow r$.
On peut citer d'autres classes polynomiales comme les formules Horn-renommables qui se réduisent à Horn par un renommage adéquat des littéraux et les formules q-Horn qui sont une généralisation des formules de Horn et des formules $2 - SAT$.

1.2.2 Résolustion de SAT

Le problème SAT a reçu beaucoup d'attention durant plusieurs années. De nombreuses techniques ont été proposées pour améliorer l'efficacité de sa résolution. Le principe de base varie d'une technique à l'autre et on peut citer : la résolution [Rob65], l'énumération [DLL62], la recherche locale [SKC93], les algorithmes génétiques [HLS02], etc. On peut distinguer deux catégories de méthodes pour résoudre SAT à savoir :

— **Méthodes incomplètes** : Elles cherchent à trouver des solutions en utilisant des heuristiques sans exploration exhaustive de l'espace de recherche. Ces méthodes ne sont pas capables de détecter l'insatisfiabilité. Lorsqu'aucune solution n'est générée après un certain temps, on ne peut pas dire si les solutions existantes ont été manquées ou si le probléme est en effet insatisfiable. La plupart de ces méthodes sont stochastiques, c-à-d, qu'elles utilisent des mouvements (affectations des variables à 0 ou 1) aléatoires. Ces méthodes ont commencé à avoir un succès considérable sur SAT dans les années 1990 avec l'introduction de $GSAT$ [SLM92] qui montre qu'une recherche locale basée sur l'objectif de maximisation du nombre de clauses satisfaites, peut être utilisée pour résoudre un grand nombre d'instances du problème SAT.
— **Méthodes complètes** : Elles explorent l'espace de recherche en entier en utilisant la recherche par retour arrière « backtrack ». Plusieurs techniques d'élagage de l'espace de recherche ont été proposées pour éviter l'exploration des sous espaces qui ne contiennent pas de solutions car l'exploration exhaustive de tout l'espace est très coûteuse.

Les solveurs SAT basés sur la recherche locale ont réalisé de grands progrès. Cependant, les solveurs SAT complets ont connu des améliorations considérables dues à l'invention de techniques d'apprentissage et de retour arrière non chronologiques [SS97, BJS97]. De plus, la vérification formelle est l'une des applications les plus importantes de SAT et dans ce type d'application les solveurs complets sont les plus performants. Nous présontons dans la suite de cette section les plus connus des algorithmes complets qui ont été proposés pour résoudre le problème SAT. Avant de détailler ces algorithmes, nous définissons quelques notions nécessaires pour la suite.

Définition 1.12 (Subsumption)**.** *On dit qu'une clause c_i subsume une clause c_j si et seulement si $c_i \subseteq c_j$ et on a $c_i \models c_j$.*

Définition 1.13 (Résolvante)**.** *Soit c_i une clause qui contient le litéral l et c_j une clause qui contient le litéral $\bar{l}$. La résolvante r est définie par la clause $r = c_i \cup c_j \setminus \{l, \bar{l}\}$ et notée $\rho(l, c_i, c_j)$.*

Proposition 1.1. *Soient $\mathcal{F}$ une formule et r la résolvante de deux clauses c_i , c_j de $\mathcal{F}$ alors :*
— *$c_i \wedge c_j \models r$.*
— *$\mathcal{F} \equiv \mathcal{F} \cup r$.*

Définition 1.14 (Propagation d'un littéral)**.** *Soit $\mathcal{F}$ une formule propositionnelle et l un littéral dans $\mathcal{L}(\mathcal{F})$. La propagation de l dans $\mathcal{F}$, notée $\mathcal{F}|_l$, est la formule obtenue en éliminant les clauses contenant l et en supprimant $\bar{l}$ de toutes les clauses le contenant. Formellement, $\mathcal{F}|_l = \{c \in \mathcal{F},\ \{l, \bar{l}\} \cap c = \emptyset\} \cup \{c \setminus \{\bar{l}\} | c \in \mathcal{F},\ \bar{l} \in c\}$.*

Définition 1.15 (Propagation unitaire)**.** *Soit $\mathcal{F}$ une formule propositionnelle. La formule obtenue par l'application de la propagation unitaire (PU) sur $\mathcal{F}$, noée $\mathcal{F}^*$, est définie récursivement comme suit :*

1. *$\mathcal{F}^* = \mathcal{F}$ si pour toute clause $c \in \mathcal{F},\ |c| \neq 1$.*
2. *$\mathcal{F}^* = \bot$ s'il exist deux clauses $c_1, c_2 \in \mathcal{F}$ tel que $c_1 = \{l\}$ et $c_2 = \{\bar{l}\}$.*
3. *$\mathcal{F}^* = (\mathcal{F}|_l)^*$ s'il existe une clause $c \in \mathcal{F}$ tel que $|c| = 1$ et $c = \{l\}$.*

Définition 1.16 (Littéral pur)**.** *Soit $\mathcal{F}$ une formule propositionnelle et $l \in \mathcal{L}(\mathcal{F})$. l est un littéral pur si et seulement si $\bar{l} \notin \mathcal{L}(\mathcal{F})$.*

Propriété 1.1. *Soient $\mathcal{F}$ une formule propositionnelle et l un littéral pur de $\mathcal{F}$. $\mathcal{F}$ est satisfiable si et seulement si $\mathcal{F}|_l$ est satisfiable.*

Algorithme DP : DP [DP60] est une méthode complète qui prend en entrée une formule sous forme clausale (CNF). Le principe fondamental de DP est d'utiliser l'opération de résolution afin de projeter l'ensemble des informations logiques de la formule initiale sur un sous-ensemble de ses variables. La résolution est alors appliquée jusqu'à élimination de toutes les variables. Si une clause vide est générée après une opération de résolution alors la formule est instatisfiable ; sinon le processus continue jusqu' à l'élimination de toutes les variables par résolution sans générer de clause vide et dans ce cas la formule est satisfiable.

L'algorithme 1 illustre le mécanisme de fonctionnement de la méthode DP. Étant donnée une formule $\mathcal{F}$, pour chaque variable choisie p, on génère toute les résolvantes entre les clauses qui contiennent le litéral positif p ($\mathcal{F}_p$) et les clauses qui contiennent le litéral négatif $\neg p$ ($\mathcal{F}_{\neg p}$) puis on remplace ces clauses par toutes les résolvantes produites (ligne 4-5). Si une des résolutions génère une clause vide (ligne 6) alors $\mathcal{F}$ est insatisfiable (ligne 7), sinon si toutes les variables ont été éliminées par résolution, alors $\mathcal{F}$ est satisfiable (ligne 10).

Le problème avec cette approche est que l'application de la résolution peut provoquer une croissance exponentielle de la formule durant le processus de génération de toutes les résolvantes possibles. De ce fait, cette technique dans sa forme originale est rarement utilisée en pratique [RD00]. Cependant, une forme limitée de DP est utilisée dans des pré-traitements de formules CNF intégrés dans la plupart des solveurs SAT modernes, à savoir la méthode « SateELite » [EB05] qui applique la résolution pour éliminer une variable uniquement si la taille de la formule n'augmente pas.

Algorithm 1: Procédure DP

Input: Une formule CNF $\mathcal{F}$
Output: $vrai$ si $\mathcal{F}$ est satisfiable, $faux$ sinon

1 $V = \mathcal{P}(\mathcal{F})$;
2 **while** $V \neq \emptyset$ **do**
3 p= ChoisirVariable(V);
4 V=$V \setminus \{p\}$;
5 $\mathcal{F} \leftarrow \mathcal{F} \setminus (\mathcal{F}_p \cup \mathcal{F}_{\neg p})$;
6 $\mathcal{F} \leftarrow \mathcal{F} \cup \rho(p, \mathcal{F}_p, \mathcal{F}_{\neg p})$;
7 **if** $\emptyset \in \mathcal{F}$ **then**
8 **return** $faux$;
9 **return** $vrai$;

Exemple 1.9. *Soit $\mathcal{F}$ la formule CNF suivante :*

$$(\neg p \vee \neg q \vee r) \wedge (p \vee \neg q) \wedge (q \vee r) \wedge (\neg p \vee f \vee e)$$

itération 1 :élimination de la variable p en appliquant la procédure DP.

$$(\neg q \vee r) \wedge (q \vee r) \wedge (\neg q \vee f \vee e)$$

itération 2 : élimination de la variable q en appliquant la procédure DP.

$$r \wedge (r \vee f \vee e)$$

itération 3 : Lélimination de la variable r en appliquant la procédure DP supprime toute les clasues ($\mathcal{F} = \emptyset$) et donc la formule est satisfiable.

Notons que nous utilisons la notation $\rho(p, \mathcal{F}_p, \mathcal{F}_{\neg p})$ pour désigner l'ensemble des résolvantes possibles entre toutes les clauses contenant p et celles contenant $\neg p$.

Algorithme $DPLL$ **:** L'algorithme de Davis, Putnam, Logemann et Loveland (nomé $DPLL$) [DLL62] est une méthode complète basée sur une recherche par retour arrière. Cette approche explore un arbre de recherche et utilise quelques techniques de raisonnement dans chaque nœud pour élaguer certaines branches qui ne contiennent pas de solution. Ces techniques sont principalement la propagation des littéraux purs et des littéraux unitaires. Dans chaque nœud un littéral de décision est choisi pour étendre l'interprétation partielle courante. Pour chaque littéral de décision il existe deux possibilités, soit le littéral est vrai, soit il est faux. Par conséquent, l'espace de recherche est décomposé en deux sous espaces à chaque nœud de décision car une formule $\mathcal{F}$ est satisfiable si et seulement si $\mathcal{F}|_p$ est satisfiable ou $\mathcal{F}|_{\neg p}$ est satisfiable. Après la propagation d'un littéral de décision, une séquence de littérraux propagés est produite en déclenchant un processus de propagations sur toutes les clauses unitaires jusqu'à ce qu'il ne reste plus de clauses unitaires. Cette façon de faire est différente de DP car cette dernière procède par élimination de variables en remplaçant le problème initial par un sous-problème plus large, alors que $DPLL$ procède par séparation. Le choix de la prochaine variable à affecter est fait suivant une heuristique (e.g. choix de la variable qui apparaît le plus souvent dans les clauses les plus courtes).

$DPLL$ est une méthode qui se base principalement sur deux points clés. La simplification qui permet la réduction de la formule en une formule plus petite est équivalente pour la satisfiabilité et l'heuristique de branchement utilisée qui permet la réduction de la taille de l'arbre de recherche à explorer. De nombreuses contributions ont été proposées pour améliorer l'algorithme $DPLL$ (algorithme 2). Elles concernent principalement les points suivants :

1. simplification de la formule ;
2. choix de la variable de décision ;
3. traitement des échecs.

L'algorithme 2 est une version récursive de $DPLL$. Cette algorithme prend en entrée une formule CNF $\mathcal{F}$ et retourne $vrai$ si la formule est satisfiable, $faux$ sinon. Après propagation de tous les littéraux unitaires (ligne 2-3), si la clause vide $\perp$ est produite alors la formule $\mathcal{F}$ est insatisfiable et l'agorithme retourne $faux$. Sinon, une variable de décision p est choisie (ligne 6) et l'espace de recherche est décomposé en deux (lignes 7-10) pour tester les deux branches possibles p et $\neg p$. Si une des deux branches retourne $vrai$ alors la formule $\mathcal{F}$ est satisfiable, sinon la formule est insatisfiable.

Exemple 1.10. *Soit $\mathcal{F}$ la formule CNF suivante :*

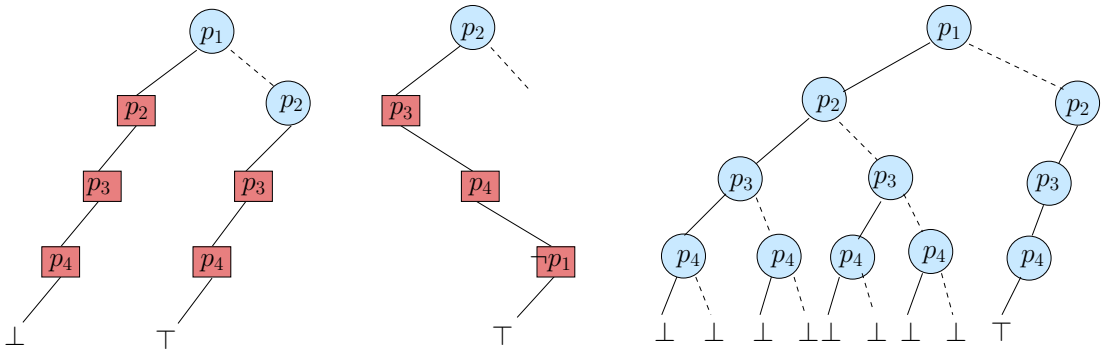
$$(\neg p_1 \vee p_2) \wedge (\neg p_2 \vee p_3) \wedge (\neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_4)$$

La figure 1.1 Présente trois arbres de recherche qui illustrent l'impacte de l'heuristique de choix de variable sur l'exploration de l'arbre de recherche dans un algorithme $DPLL$ (les figures (a) et (b)) et l'importance de la propagation unitaire dans une procédure $DPLL$ comparée à une procédure avec retour arrière chronologique et sans propagation unitaire. En effet, dans le cas de $DPLL$ (figure (b)) une décision p_2 suivie de la propagation unitaire est suffisante pour trouver un modèle alors que dans la figure (a) il a fallu réfuter la décision p_1 est décider d'affecter p_2 à vrai pour trouver le modèle. Dans le cas de l'algorithme avec retour arrière chronologique (figure (c)), un sous-arbre a été développé avant de réfuter la décision d'affecter p_1 à $vrai$.

Algorithm 2: Algorithme $DPLL$

Input: Une formule CNF $\mathcal{F}$
Output: $vrai$ si $\mathcal{F}$ est satisfiable, $faux$ sinon

```
1  DPLL(F){
2  if ∃c ∈ F, | c |= 1 et c = {p} then
3  |  return DPLL(F|_p);
4  if ∃c ∈ F, c = ⊥ then
5  |  return faux;
6  p ← ChoisirVariable(F);
7  if DPLL(F|_p) = vrai then
8  |  return vrai ;
9  else
10 |  return DPLL(F|_¬p);
11 }
```



(a) (b) (c)

FIGURE 1.1 – Propagation unitaire et choix de variables dans un algorithme $DPLL$

Techniques de simplification : Les techniques de simplification d'une formule sont très importantes car elles permettent la réduction de l'espace de recherche en évitant l'exploration de certaines branches ce qui accélère la recherche et réduit le temps de calcul. Deux types de simplification ont été introduites. Des simplifications avant le début de la recherche appelées prétraitements et des simplifications au cours de la recherche. Parmi ces techniques on peut citer comme exemples :

— *Règle des littéraux purs* [DLL62] : Si une variable apparaît positivement (resp. négativement) dans toutes les clauses alors cette variable peut être affectée à vrai (resp. faux) (voir la propriété 1.1).
— *Traitements locaux par propagation unitaire* : Ils ont été proposés dans CSAT [DABC93]. Ces traitements permettent de produire pendant la résolution des littéraux impliqués ou de détecter des incohérences locales.
— *Prétraitement par résolution* : La résolution est l'opération de base utilsée dans la plupart des prétraitements. Plusieurs contributions explorent la possibilité de calculer des résolvantes soit pour enrichir le problème avec des clauses redondantes ou pour le simplifier. Dans [SP05] les auteurs proposent un système, appelé NiVer, qui élimine des variables par résolution sans augmenter la taille de la formule. Bacchus et Winter [BW03] ont exploré l'utilisation d'une variante

de la résolution, appelée hyper-résolution, pour simplifier la formule. SatElite [EB05] utilise la résolution et la détection de subsumptions pour supprimer des clauses et des littéraux. Finalement, les auteurs de ReVival [PHS08] proposent un traitement de redondances par propagation unitaire. Le test de redondance par propagation unitaire d'une clause est capturé par un graphe d'implication qui est lui même utilisé pour déduire une sous-clause.

— *Exploitation des classes polynomiales :* de nombreux problèmes réels contiennent des contraintes qui sont encodées par des clauses binaires ou des clauses de Horn. Par conséquent, plusieurs contributions ont été proposées pour exploiter ces classes au cours de la résolution comme [Lar92, BHS94].

Les techniques de simplification permettent de réduire le nombre de branches à explorer durant la recherche. Mais malheureusement, la plupart d'entre elles sont coûteuses à mettre en œuvre et réduisent l'efficacité globale du solveur pour les instances SAT générales bien qu'elles soient utiles pour résoudre certaines classes d'instances. Trouver un bon compromis entre des algorithmes rapides qui calculent des déductions simples et des méthodes de raisonnement plus sophistiquées, mais plus lentes, est une préoccupation centrale qui guide la recherche sur les solveurs SAT.

Heuristiques de branchement : La relation forte entre l'ordre d'affectation des variables et la taille de l'arbre de recherche développé a augmenté l'importance des heuristiques de branchement. En effet, sélectionner la bonne variable et la bonne valeur de cette dernière est une étape cruciale pour un solveur SAT. Les heuristiques utilisées pour choisir les valeurs sont généralement basées sur de simples statistiques. En pratique, la plupart des instances SAT difficiles sont insatisfiables et le solveur doit explorer tout l'espace de recherche. Par conséquent, le principal objectif des heuristiques de branchement est de découvrir les conflits le plus tôt possible. Une bonne heuristique de branchement dans SAT ne doit pas être coûteuse à évaluer. Par exemple, une heuristique qui nécessite l'itération sur toutes les clauses du problème ne serait clairement pas abordable sur de grandes instances. Les heuristiques de branchement les plus réussies ont toutes une complexité de temps très réduite par rapport à la taille de la formule.

— *Heuristique* $MOMS$ « Maximum Occurrences in Clauses of Minimum Size » : C'est la plus connue des heuristiques de sélection de variable de branchement. Elle est associée à $DPLL$ et introduite dans [DLL62, GOL79]. Elle sélectionne la variable ayant le plus d'occurrences dans les clauses de plus petites tailles. Ce choix se justifie par le fait qu'elle favorise la propagation unitaire. Il est à noter qu'une amélioration de cette heuristique a été proposée dans [DABC93] pour équilibrer les sous-arbres produits par l'affectation d'une variable.

— *Heuristique JW* : L'heuristique de branchement Jeroslow-Wang (JW) [JW90] est également basée sur la fréquence d'apparition d'une variable dans les clauses les plus petites. De plus, elle associe un critère d'équilibrage de la taille des deux sous-arbres résultants de l'opération de division. Les variables ayant un critère de représentativité restreint à leur forme négative proche de celui restreint à leur forme positive sont privilégiées. Afin de satisfaire au mieux les deux critères (fréquence d'apparition et équilibre des signes) le calcul suivant est proposé :

$$JW(p) = \alpha * m(p) * m(\overline{p}) + m(p) + m(\overline{p}) + 1 \tag{1.3}$$

Où $m(p) = (\sum_{c \in \mathcal{F}, p \in c} 2^{-|c|})$ (resp. $m(\overline{p})$) représente la mesure de représentativité du littéral p (resp. $\overline{p}$) et α est une constante empiriquement fixée. La variable p sélectionnée par l'heuristique JW est celle qui maximise $JW(p)$.

— *Heuristique* $BOHM$: Cette heuristique a été proposée par Buro et al dans [BB92]. Elle est légèrement différente de $MOMS$ dans sa mesure de représentativité mais l'idée principale reste la même et consiste à privilégier les variables ayant un maximum d'occurrences dans les clauses

les plus courtes. Cette heuristique permet une discrimination plus fine en associant à chacune des n variables de la formule le vecteur ordonné suivant :

$$S_{BOHM}(p) = (H_1(p), H_2(p), ..., H_n(p))$$

Où $H_k(p)$ représente le nombre d'apparitions du littéral p dans les clauses de longueur k. $H_k(p)$ est estimé comme suit :

$$max(H_k(p), H_k(\overline{p})) + 2 * min(H_k(p), H_k(\overline{p})) \tag{1.4}$$

La variable choisie est celle qui maximise le vecteur $S_{BOHM}(p)$ suivant l'ordre lexicographique. Par exemple si $\forall p \neq q, H_1(p) > H_1(q)$ alors la variable p est choisie, sinon, en cas d'égalité nous passons à H_2 et ainsi de suite.

— *Heuristique* UP : Elle est basée sur l'heuristique $MOMS$ avec une particularité intéressante d'exploiter la propagation unitaire pour sélectionner une variable en prévoyant à l'avance son influence sur le processus de la recherche [LA97]. La procédure de propagation unitaire est utilisée pour étudier l'influence d'une variable non affectée p. Pour une formule $\mathcal{F}$, l'estimation du poids de la variable p est effectuée par le calcul de la réduction de la taille de la formule sur les deux branches $(\mathcal{F}|_p)^*$ et $(\mathcal{F}|_{\overline{p}})^*$. Ce traitement favorise d'une part l'apparition des futures propagations unitaires et d'autre part la détection des incohérences locales. Cette heuristique est implémentée dans l'ensemble des solveurs basés sur le « look-ahead » [LA97, DD06, ZMMM01]. Toutefois, bien qu'elle soit plus efficace que $MOMS$, le coût de calcul de UP est relativement élevé.

Analyse de conflits : Dans une approche $DPLL$, un retour arrière chronologique (retour vers la décision $n-1$) est effectué après chaque détection de conflit sans prise en compte des raisons qui ont mené à ce conflit. L'analyse de conflit permet d'effectuer des retours arrières non chronologiques d'un nœud de décision de niveau n à un nœud de décision de niveau $m < n$ en tenant compte des décisions à l'origine du conflit. Ceci permet d'éviter de redécouvrir les mêmes échecs. L'analyse de l'échec est une approche qui a été utilisée dans plusieurs domaine de l'intelligence artificielle, à savoir les systèmes de maintien de vérité [McA80, SS77], les problèmes de satisfaction de contraintes [Dec90, Gin93, SV94] et la programmation logique [Bru80]. Ces approches se distinguent essentiellement par les techniques utilisées pour analyser les échecs et éviter de rencontrer les mêmes conflits au cours de la recherche. Étant donné une interprétation conflictuelle $\mathcal{I}$, l'analyse de conflit permet d'identifier le plus petit sous-ensemble de décisions $C \subset \mathcal{I}$ qui a causé le conflit [BJS97]. À partir de cet ensemble, appelé l'ensemble conflit, la dernière décision dans la branche qui appartient à C est déterminée et un retour arrière non chronologique vers cette décision est par la suite effectué pour changer la valeur de vérité de cette variable. Après un retour arrière non chronologique, le conflit est évité partiellement car les mêmes conflits peuvent réapparaître par la suite dans ce qui reste de la recherche. Pour éviter la reproduction des mêmes erreurs, des clauses sont ajoutées pour représenter chaque conflit. À chaque fois qu'une contradiction est détectée, une clause impliquée par la formule est identifiée permettant à la propagation unitaire de réaliser de nouvelles propagations. Cette clause est appelée clause apprise « conflict-driven clause » [SS96, ZMMM01, BKS04, BJS97]. L'ajout de toutes les clauses apprises à la base permet de ne pas répéter les mêmes conflits durant le reste de la recherche.

Exemple 1.11. *Soit $\mathcal{F}$ la formule CNF suivante :*

$$\begin{array}{lll}
(c_1) & (p_1 \vee p_2) & \wedge \\
(c_2) & (p_2 \vee p_3) & \wedge \\
(c_3) & (\neg p_1 \vee \neg p_4 \vee p_5) & \wedge \\
(c_4) & (\neg p_1 \vee p_4 \vee p_6) & \wedge \\
(c_5) & (\neg p_1 \vee \neg p_4 \vee p_6) & \wedge \\
(c_6) & (\neg p_1 \vee p_4 \vee \neg p_6) & \wedge \\
(c_7) & (\neg p_1 \vee \neg p_5 \vee \neg p_6) &
\end{array}$$

Après avoir affecté les littéraux p_1, p_2, p_3 et p_4, la propagation unitaire permet de propager les littéraux p_5 (clause c_3) et p_6 (clause c_5) et la clause c_7 devient vide (tous les littéraux sont à 0). Dans ce cas, l'ensemble conflit est l'ensemble des littéraux qui ont mené au conflit $\{p_1,\ p_4,\ p_5,\ p_6\}$. Notons que p_2 et p_3 ne participent pas au conflit. la clause apprise est alors $(\neg p_1 \vee \neg p_4)$ et l'algorithme teste ensuite l'autre branche $\neg p_4$. L'algorithme détecte un nouveau conflit et l'ensemble conflit est $\{p_1, \neg p_4, p_6\}$. Après le test exhaustive des deux valeurs possibles de la variable p_4, le retour-arrière non chronologique peut maintenant se faire au niveau de p_1 qui doit être affectée à 0 et le processus est réitéré.

Solveurs SAT modernes : Les solveurs SAT modernes sont particulièrement efficaces avec les instances SAT structurées provenant d'applications industrielles. Gomes et al [GSCK00] ont observé que sur les instances industrielles, de différents ordres de variables conduisent souvent à des différences importantes dans le temps de résolution. Cette observation explique d'un côté l'introduction de politiques de redémarrage dans les solveurs SAT modernes, qui tentent de découvrir un bon ordre des variables. D'un autre côté, l'heuristique $VSIDS$ « Variable State Independent Decaying Sum » et d'autres variantes d'heuristiques basées sur l'activité ont été introduites pour concentrer la recherche sur les parties les plus contraignantes de la formule. Les redémarrages et $VSIDS$ jouent des rôles complémentaires puisqu'ils mettent en œuvre les deux principes de diversification et intensification. L'apprentissage de clauses à partir d'un conflit ($CDCL$: « Conflict Driven Clause Learning ») est la troisième composante, conduisant à un retour en arrière non-chronologique. L'activité des variables est mise à jour au cours du processus d'apprentissage, permettant toujours à $VSIDS$ de sélectionner la variable la plus active en tant que nouveau point de décision. Comme le nombre de clauses apprises peut être exponentiel dans le pire des cas, plusieurs stratégies de suppression sont proposées pour conserver une base de clauses apprises de taille gérable. Pour permettre la résolution de très grandes instances SAT, l'introduction d'une structure de données paresseuse « watched literals » est cruciale pour l'efficacité des solveurs SAT modernes. Finalement, réduire la taille de la formule d'entrée dans une étape de prétraitement est une autre composante très importante intégrée dans une majorité des solveurs SAT.

1. *Stratégies de redémarrage* : La politique de redémarrage a été introduite dans [GSK98]. Lorsqu'un solveur redémarre, l'interprétation partielle actuelle est abandonnée et la recherche recommence à la racine de l'arbre de recherche, tout en conservant plusieurs informations cumulées à partir des exécutions précédentes (par exemple les clauses apprises, les activités des variables). L'argument majeur derrière l'adoption du redémarrage est l'observation qui démontre qu'en utilisant plusieurs ordres de variables, l'efficacité des solveurs peut varier de plusieurs ordres de grandeur. Le but des redémarrages est d'éliminer les longues exécutions en augmentant la probabilité de tomber sur les meilleures ordres de variables. De nombreuses stratégies de redémarrage ont été proposées. Parmi les plus connues on trouve la série arithmétique paramétrée par x, y qui est une stratégie où un redémarrage est effectué après chaque x conflits et qui augmente de y après chaque redémarrage [Hua07]. La série géométrique parametrée par x, y est une autre stratégie dans laquelle un redémarrage est effectuée chaque x conflits et la valeur de x se multiplie par un facteur y à chaque redémarrage. Enfin, la série Luby [LSZ93, RS08] paramérée par x est

une stratégie qui évolue grâce à la suite de nombres t_i multipliés par la constante x tel que :

$$t_i = \begin{cases} 2^{k-1} & si\ \exists k \in N,\ i = 2^k - 1 \\ t_{i-2^{k-1}+1} & si\ \exists k \in N,\ 2^{k-1} \leqslant i \leqslant 2^k - 1 \end{cases}$$

La plupart des stratégies de redémarages sont statiques et la valeur de coupure suit différents schémas d'évolution (par exemple arithmétique, géométrique, Luby). En pratique, les redémarrages rapides fonctionnent bien sur les instances industrielles, mais sur les instances SAT difficiles, les redémarrages lents sont plus appropriés. Récemment, plusieurs politiques de redémarrages dynamiques ont été proposées et la valeur de coupure est calculée durant la recherche en utilisant des informations observées sur le comportement du solveur durant la recherche [HJS10, PD09].

2. *Heuristiques basées sur l'activité* : La plupart des heuristiques de choix de variables présentées précédemment se reposent sur des arguments syntaxiques tels que le nombre d'occurrences des variables et la taille des clauses. Avec l'évolution des solveurs SAT et l'apparition de l'analyse de conflit, un autre type d'heuristiques dynamique est apparu. Ces heuristiques sont dirigées par les conflits indépendamment de l'interprétation courante avec un coût de calcul négligeable. L'heuristique $VSIDS$ proposée dans [ZMMM01] associe une activité à chaque variable. Cette dernière est augmentée à chaque fois que la variable associée apparaît dans une clause apprise à partir d'un conflit. De plus, l'activité de toutes les variables est multipliée par une constante inférieure à 1 pour favoriser les variables qui sont récement impliquées dans des clauses apprises. Une idée similaire est utilisées dans $MiniSAT$ [Een05]. Cependant, la mise à jour de l'activité n'est pas limitée aux variables apparaissant dans la clause apprise du conflit, mais il considère toutes les variables apparaissant dans n'importe quelle clause utilisée dans l'analyse de conflit. Dans [BGS99], les auteurs proposent une extension du schéma de pondération des clauses aux techniques de type $DPLL$. À Chaque fois qu'un conflit est atteint, l'activité des clauses falsifiées est augmentée. La variable suivante est choisie en fonction de ses occurrences dans la plupart des clauses falsifiées. Les heuristiques $VSIDS$ et celle de $MiniSAT$ sont peu coûteuses à maintenir et elles se révèlent très efficaces sur une variété de problèmes.

3. *Structures de données paresseuses « watched literals »* : La propagation unitaire consomme 80% du temps de calcul dans les solveurs modernes ce qui a poussé à l'optimisation des structures de données afin d'augmenter l'efficacité de la propagation unitaire. Les auteurs de [ZMMM01] ont constaté qu'un littéral ne peut être propagé tant qu'au moins deux littéraux de cette clause ne sont pas affectés. Par conséquent, ils ont proposé de garder deux pointeurs sur deux littéraux de chaque clause, appelés watched literals. Si l'un de ces littéraux est affecté à $faux$ le pointeur est déplacé vers un autre littéral qui n'est pas à $faux$. Lorsque les deux littéraux pointent le même littéral, ce littéral est unitaire et pourra être propagé. L'intérêt de ces structures est qu'elles permettent de capturer la propagation unitaire et de ne visiter que les clauses concernées par les affectations. De plus, aucune mise à jour n'est nécessaire lors du retour-arrière. Pour plus de détails sur le fonctionnement de ces pointeurs voir [Een05, MMZ+01].

4. *Stratégies de suppression des clauses apprises* : L'apprentissage de clauses est reconnu comme une technique puissante à la fois en théorie et en pratique. Cependant, l'ensemble des clauses qui peuvent être dérivées des conflits est de taille exponentielle dans le pire des cas. En pratique, le nombre de telles clauses peut parfois être supérieur au nombre de clauses de la formule originale. Pour avoir une propagation unitaire de coût raisonnable, il est important de maintenir une base de clauses apprises de taille polynomiale. Pour cette raison, plusieurs stratégies ont été proposées pour réduire dynamiquement la base des clauses apprises en supprimant des clauses considérées comme non pertinentes aux étapes de recherche suivantes. Dans la plupart des cas, ces stratégies préfèrent conserver les clauses les plus petites, récentes et actives. Par exemple, dans [ALMS11]

Algorithm 3: Solveur $CDCL$

Input: Formule CNF $\mathcal{F}$
Output: un modèle de $\mathcal{F}$ ou $unsat$ si $\mathcal{F}$ est insatisfiable

```
1  D ← ∅;                                          /* littéraux de décision */
2  Δ ← ∅;                                        /* base des clauses apprises */
3  while (vrai) do
4      S ← F ∧ Δ ∧ D;
5      if (S* = ⊥) then
6          if ((D = ⟨⟩)) then
7              return unsat ;
8          α ← learningFromConflict(S) ;
9          m ← assertion level of α;
10         D ← D^m;
11         Δ = Δ ∧ α ;
12     else
13         if (timeToReduce()) then
14             Δ ← reduceLearntDB();
15         if (timeToRestart()) then
16             D ← ∅;
17             S ← F ∧ Δ;
18         ℓ ←decide();
19         if (ℓ = null) then
20             return D;
21         D ← D ∧ ℓ;
```

les auteurs proposent une politique basée sur un principe dynamique de gel et d'activation des clauses apprises. A un état de recherche donné, elle active les clauses les plus prometteuses tout en gelant les clauses jugées non pertinentes. De cette façon, les clauses précédemment apprises peuvent être écartées dans l'étape en cours, mais peuvent être réactivées dans les étapes ultérieures du processus de recherche. Cette politique tente d'exploiter des informations collectées dans le passé pour déduire la pertinence d'une clause donnée pour les étapes de recherche restantes.

5. *Solveur* $CDCL$: L'algorithme 3 présente les étapes de base d'un solveur $CDCL$. L'algorithme commence par un ensemble vide de littéraux de décision et une base de clauses apprises vide (lignes 1 et 2). A chaque itération, la propagation unitaire est appliquée sur la formule actuelle $\mathcal{S}$ (ligne 5). En cas de conflit (lignes 6-11), si l'ensemble de littéraux de décisions est vide (ligne 6), la formule est insatisfiable, sinon une nouvelle clause est dérivée après une analyse du conflit (ligne 8). Dans ce dernier cas, l'algorithme effectue un retour arrière vers le niveau de décision m (ligne 9) et ajoute la clause dérivée à la base des clauses apprises (ligne 11). Si aucun conflit ne s'est produit (lignes 12-21), l'algorithme réduit la base (lignes 13-14) et/ou redémarre le processus de recherche (lignes 15-17) en utilisant une politique de réduction et de redémarrage respectivement. Ensuite, un nouveau littéral de décision est sélectionné en utilisant une heuristique $VSIDS$ (ligne 18). Le littéral choisi est ensuite ajouté à l'ensemble des décisions (ligne 21). Si toutes les variables sont affectées (ligne 19-20), alors un modèle est trouvé et la formule est satisfiable.

1.2.3 Encodage des contraintes de cardinalité en SAT

Les améliorations importantes des solveurs SAT ont donné lieu à l'élargissement de la classe de problèmes du monde réel qui peuvent être résolus dans la pratique. La phase de modélisation d'un tel nombre croissant d'applications plus complexes vers des formules propositionnelles en forme normale conjonctive (CNF) devient encore plus cruciale. Le problème de modélisation suit plusieurs transformations polynomiales et étapes de réécriture, en commençant par une description de haut niveau, en utilisant un langage de haut niveau ou une logique propositionnelle complète, jusqu'à la formulation de bas niveau, généralement une formule CNF. L'ensemble du processus préserve la satisfiabilité propositionnelle, grâce au principe d'extension [Tse68], permettant l'introduction de nouvelles variables pour représenter des sous-formules ou des contraintes complexes. Parmi ces contraintes, les contraintes de cardinalité et pseudo-booléennes, exprimant des bornes numériques sur des quantités discrètes, sont les plus populaires car elles apparaissent fréquemment dans l'encodage de nombreux problèmes du monde réel : ordonnancement, vérification formelle, configuration de produits et fouille de données. Pour ces raisons, de nombreuses approches ont été proposées pour trouver un encodage efficace de la contrainte de cardinalité (par exemple [War98, BB03, Sin05, MSL07, ANORC09, JSS14]) et des contraintes pseudo-booléennes (par exemple [ES06, BBR09]) vers une formule CNF. L'efficacité de ces encodages est mesurée par la compacité de la représentation (taille de la formule CNF) et la capacité d'atteindre le même niveau de propagation de contraintes (cohérence d'arc généralisée) sur la formule CNF. Nous citons comme exemples les trois encodages suivants :

— *Encodage basé sur le probème des pigeons* : Dans [JSS14], les auteurs ont proposé un nouvel encodage de la contrainte de cardinalité $\sum_{i=1}^{n} x_i \geqslant k$ basé sur le probème des pigeons. Ils ont observé que la sémantique de la contrainte de cardinalité peut être exprimée de la même façon que le problème de mettre k pigeons dans n pigeonniers. La première formulation, appelée ${\mathcal{P}_n}^k$, est simplement exprimée par l'ensemble de contraintes suivant :

$$\bigwedge_{j=1}^{k} (\neg p_{ji} \vee x_i), \quad 1 \leqslant i \leqslant n \tag{1.5}$$

$$\bigvee_{i=1}^{n} p_{ji}, \quad 1 \leqslant j \leqslant k, \tag{1.6}$$

$$\bigwedge_{1 \leqslant j < j' \leqslant k} (\neg p_{ji} \vee \neg p_{j'i}), \quad 1 \leqslant i \leqslant n \tag{1.7}$$

Les contraintes (1.6) et (1.7) encodent le problème des pigeons ${PHP_n}^k$, où k est le nombre de pigeons et n est le nombre de pigeonniers (p_{ji} exprime que le pigeon j est dans le pigeonnier i). Malheureusement, la vérification de la satisfiabilité d'un problème de Pigeons est difficile. Pour maintenir la cohérence d'arc généralisée (GAC), les auteurs ont proposé une amélioration obtenue en cassant les symétries entre les variables p_{ij} impliquées dans l'expression du problème de pigeons (contraintes (1.6) et (1.7)) par résolution entre les clauses des prédicats d'élimination de symétries et celles de ${\mathcal{P}_n}^k$. Les auteurs ont dérivé l'encodage suivant, appelé $ph{\mathcal{P}_n}^k$:

$$\bigwedge_{1 \leq i \leq k} \Big(\bigvee_{1 \leq j \leq n-k+1} p_{ij} \Big) \tag{1.8}$$

$$\bigwedge_{1 \leq i \leq k} \bigwedge_{1 \leq j \leq n-k+1} (x_{i+j-1} \vee \neg p_{ij}) \tag{1.9}$$

$$\bigwedge_{1\leq i<k}\bigwedge_{1\leq j<n-k+1}(\neg p_{(i+1)j}\vee\bigvee_{1\leq l\leq j}p_{il}) \tag{1.10}$$

Exemple 1.12. *Considérons l'inéquation $x_1+x_2+x_3+x_4+x_5+x_6\geqslant 4$. En utilisant l'encodage basé sur le problème des pigeons phP_6[4], on obtient la formule CNF suivante :*

$p_{11}\vee\neg p_{21}$ $\quad x_1\vee\neg p_{11}$ $\quad x_2\vee\neg p_{12}$

$p_{11}\vee p_{12}\vee\neg p_{22}$ $\quad x_2\vee\neg p_{21}$ $\quad x_3\vee\neg p_{13}$ $\quad p_{11}\vee p_{12}\vee p_{13}$

$p_{21}\vee\neg p_{31}$ $\quad x_3\vee\neg p_{22}$ $\quad x_3\vee\neg p_{31}$ $\quad p_{21}\vee p_{22}\vee p_{23}$

$p_{21}\vee p_{22}\vee\neg p_{32}$ $\quad x_4\vee\neg p_{23}$ $\quad x_4\vee\neg p_{32}$ $\quad p_{31}\vee p_{32}\vee p_{33}$

$p_{31}\vee\neg p_{41}$ $\quad x_4\vee\neg p_{41}$ $\quad x_5\vee\neg p_{33}$ $\quad p_{41}\vee p_{42}\vee p_{43}$

$p_{31}\vee p_{32}\vee\neg p_{42}$ $\quad x_5\vee\neg p_{42}$ $\quad x_6\vee\neg p_{43}$

— *Encodage basé sur les réseaux de tri « Sorting networks »* : L'un des encodages les plus efficaces pour les contraintes de cardinalité est l'encodage basé sur les réseaux de tri [ES06]. Dans cet encodage, la contrainte de cardinalité $\sum_{i=1}^{n}x_i\leqslant k$ est traduite en un seul trieur de n entrées $X=\{x_1,...,x_n\}$ et de n sorties $Z=\{z_1,...,z_n\}$ triées dans un ordre décroissant où la sortie $k+1$ est forcée à faux. L'idée derrière cet encodage est de trier les variables d'entrées de telles sorte que les variables vraies seront en premiers, suivies par les variables fausses. Par conséquent, la contrainte $\sum_{i=1}^{n}x_i\leqslant k$ est satisfaite si et seulement si la variable z_{k+1} est à faux. Dans [ES06], les auteurs ont prouvé que l'encodage basé sur les réseaux de tri préserve la cohérence d'arc généralisée. Notons $\Phi_{SN}^{n,k}(X;Z)$ la formule CNF qui représente le circuit basé sur les réseaux de tri qui prend comme entrée l'ensemble de variables propositionnelles X et donne comme sortie un nombre unaire représenté par l'ensemble des variables propositionnelles Z. La formule suivante définit l'encodage de la contrainte de cardinalité :

$$\Phi_{SN}^{n,k}(X;Z)\wedge\neg z_{k+1} \tag{1.11}$$

Comme les sorties Z sont triées par ordre décroissant, En fixant z_{k+1} à faux alors toutes les variables restantes $z_{k+2},...,z_n$ doivent être propagées à faux. Notons que la formule $\Phi_{SN}^{n,k}(X;Z)$ qui encode le réseau de tri est une formule de Horn, dérivée en utilisant un comparateur de base entre deux variables propositionnelles [ES06]. Le comparateur de deux variables x_1,x_2, noté $2-comp(x_1,x_2;y_1,y_2)$, est défini par la formule suivante :

$$(x_1\rightarrow y_1)\ \wedge\ (x_2\rightarrow y_1)\ \wedge\ (x_1\wedge x_2\rightarrow y_2) \tag{1.12}$$

Cette formule permet de trier les deux variables x_1 et x_2 en deux autres variables y_1, y_2 dans un ordre décroissant. Par exemple, lorsque x_1 (resp. x_2) est affectée à la valeur faux (resp. vrai), la variable de sortie y_1 (resp. y_2) est affectée à $vrai$ (resp. $faux$). Pour plus de détails, nous référons le lecteur à [ES06, ANORC11].

— *Encodage basé sur le compteur unaire séquentiel* : Cet encodage de la contrainte de cardinalité proposé par Carsten Sinz dans [Sin05] est un autre encodage très connu qui préserve la propriété de cohérence d'arc généralisée. Il calcule pour chaque variable propositionnelle x_i, la somme partielle $\sum_{j=1}^{i}x_j$ jusqu'à la valeur finale $i=n$. Les valeurs de toutes les sommes sont représentées

par des nombres unaires de taille égale à k. L'encodage est défini comme suit :

$$(\neg x_1 \vee p_{1,1}) \tag{1.13}$$

$$\bigwedge_{1<j\leqslant k} \neg p_{1,j} \tag{1.14}$$

$$\bigwedge_{1<i<n} (\neg x_i \vee p_{i,1}) \wedge (\neg p_{i-1,1} \vee p_{i,1}) \tag{1.15}$$

$$\bigwedge_{1<i<n} \bigwedge_{1<j\leqslant k} (\neg x_i \vee \neg p_{i-1,j-1} \vee p_{i,j}) \wedge (\neg p_{i-1,j} \vee p_{i,j}) \tag{1.16}$$

$$\bigwedge_{1<i\leqslant n} (\neg x_i \vee \neg p_{i-1,k}) \tag{1.17}$$

Les variables $p_{i,j}$ désignent le $j^{ème}$ chiffre de la $i^{ème}$ somme partielle p_i en représentation unaire. Les contraintes (1.13) et (1.14) correspondent au cas $i = 1$. La formule (1.17) est très importante car elle permet la détection de l'incohérence et préserve la propriété GAC en même temps. Les autres contraintes permettent la propagation de tout changement d'une somme partielle p_i après toute assignation d'une variable x_i. Notons que la formule dérivée par l'encodage basé sur le compteur unaire séquentiel est aussi une formule de Horn.

Considérons maintenant le cas particulier très connu de la contrainte de cardinalité avec $k = 1$, appelée contrainte AtmostOne. Plusieurs encodages ont été proposés pour la contrainte $\sum_{i=1}^{n} x_i \leqslant 1$ et on peut citer comme exemples :

— *Encodage par paire « pairwise encoding »* : L'encodage par paire est obtenu en considérant l'ensemble de toutes les clauses binaires négatives construites sur l'ensemble des variables $\{x_1, \ldots, x_n\}$ comme décrit dans la formule (1.18).

$$\bigwedge_{1\leq i<j\leq n} (\neg x_i \vee \neg x_j) \tag{1.18}$$

Cette formulation naïve maintient la cohérence d'arc généralisée et elle est en $\mathcal{O}(n)$ variables et $\mathcal{O}(n^2)$ clauses.

— *Encodage basé sur un compteur séquentiel et encodage basé sur le problème de Pigeons* : Le deuxième encodage de la contrainte AtMostOne est représenté par la formule (1.19) obtenue en utilisant un compteur séquentiel [Sin05]. Les auteurs de [JSS14] ont montré que le même encodage est obtenu en utilisant l'encodage basé sur le probème de pigeons décrit ci-dessus et en appliquant une étape supplémentaire d'élimination de variables par résolution. Contrairement à l'encodage par paires (1.18), celui obtenu en utilisant le compteur séquentiel (1.19) est linéaire ($\mathcal{O}(n)$ variables et clauses) grâce aux variables supplémentaires $\{p_1, \ldots, p_{n-1}\}$. Les deux encodages (1.18) et (1.19) maintiennent de la cohérence d'arc généralisée.

$$\begin{aligned}&(\neg x_1 \vee p_1) \wedge (\neg x_n \vee \neg p_{n-1}) \wedge \\ &\bigwedge_{1<i<n} (\neg x_i \vee p_i) \wedge (\neg p_{i-1} \vee p_i) \wedge (\neg x_i \vee \neg p_{i-1})\end{aligned} \tag{1.19}$$

1.2.4 Problèmes autour de SAT

Vu l'importance du probème SAT et son utilisation dans différants domaines, d'autres problèmes qui gravitent autour de SAT ont été définis pour élargir ce champ d'application. Bien que ces problèmes soient généralement beaucoup plus difficiles que SAT, ils bénéficient régulièrement des résultats obtenus dans le cadre SAT.

Définition 1.17. *Étant donnée une formule CNF $\mathcal{F}$, le problème de satisfiabilité maximum, noté MaxSAT($\mathcal{F}$), consiste à déterminer le nombre maximum de clauses de $\mathcal{F}$ qui peuvent être satisfaites.*

Notons que si une formule $\mathcal{F}$ composée de n clauses est satisfiable alors le nombre de clauses satisfaites est n. On peut noter aussi que le problème MaxSAT est NP-difficile même pour les instances polynomiales pour la satisfiabilité comme les formules 2-SAT.

MaxSAT pondéré « Weighted MaxSAT » est un autre problème d'optimisation associé à SAT. Ce problème est une généralisation de MaxSAT où un poids est associé à chaque clause. L'objectif est de maximiser la somme des poids des clauses satisfaites.

Dans certains problèmes, il y a des clauses qui doivent être satisfaites « hard clauses » et d'autres qui peuvent être satisfaites ou non « soft clauses ». Trouver une interprétation qui satisfait la partie dure et maximise le nombre de clauses satisfaites dans la partie souple est appelé MaxSAT partiel « Partial MaxSAT ». Dans d'autres probèmes, les clauses dans la partie souple n'ont pas la même importance. Des poids sont associés aux clauses souples, où le poids représente la pénalité à falsifier la clause. Par conséquent, le problème MaxSAT partiel pondéré « Weighted Partial MaxSAT » consiste à trouver l'interprétation qui satisfait les clauses dures, et minimise la somme des poids des clauses falsifiées.

Un autre problème lié à SAT est le problème d'énumération de modèles d'une formule CNF. L'énumération nécessite la génération de tous les modèles d'une instance SAT sans doublons. L'énumération de modèles est associée au problème de comptage #SAT qui consiste à déterminer le nombre de modèles pour une formule propositionnelle donnée. Le comptage de modèles est le problème canonique $\#P$-complet. Sur le plan pratique, « SampleCount » est une méthode efficace pour le comptage de modèles. C'est une approche basée sur l'échantillonnage proposée par Gomes et al. [GHSS07]. Elle permet d'avoir de très bonnes bornes inférieures avec une confiance élevée. Elle arrive à répondre à de nombreux problèmes qui sont complètement hors de portés des meilleures méthodes de comptage exactes. De même, un algorithme efficace d'énumération de modèles a été proposé dans Jabbour et al. [JLSS14].

1.3 Problème de satisfaction de contraintes

Dans cette section, nous présentons des concepts généraux liés à la modélisation des problèmes en programmation par contraintes (CP). La programmation par contraintes est un puissant paradigme utilisé pour la modélisation et la résolution des problèmes de recherche combinatoire. CP se base sur deux principes : (1) les utilisateurs modélisent le problème à résoudre en un problème de satisfaction de contraintes. (2) les solveurs recherchent les solutions. Dans cette section nous nous focalisons sur les notions qui permettent de comprendre la première étape. Pour plus de détails sur les méthodes de résolution nous orientons le lecteur vers le livre édité par Christophe Lecoutre [Lec13].

Soit X une variable. Le domaine de X, noté par Dom(X), est l'ensemble des valeurs possibles qui peuvent être assignées à X. Pour chaque variable X, seulement une seule valeur de Dom(X) est assignée

à X. Il est à noter que contrairement à SAT, les domaines des variables dans CP ne sont pas forcément binaires.

Définition 1.18 (Assignation). *Soit $\mathcal{V} = \{X_1, \ldots, X_n\}$ un ensemble de n variables. Une assignation est un tuple $(v_1, \ldots, v_n)$ de taille n tel que $v_i \in Dom(X_i)$ pour tout $i \in \{1, \ldots, n\}$.*

Exemple 1.13. *Soit X_1, X_2 deux variables avec Dom(X_1)=$\{0, 1\}$ et Dom(X_2)=$\{1, 3, 5\}$. Le tuple (0,5) est une assignation des variables X_1 et X_2. Il est équivalent à $X_1 = 0$, $X_2 = 5$.*

Définition 1.19 (Contrainte). *Soit $\mathcal{V} = \{X_1, \ldots, X_n\}$ un ensemble de n variables. Une contrainte C_i sur un ensemble de variables, noté Var(C_i)=$\{X_{i1}, \ldots, X_{ik}\} \subseteq \mathcal{V}$, est un sous-ensemble du produit cartésien des domaines des variables de Var(C_i), c-à-d, $C_i \subseteq Dom(X_{i1}) \times \ldots \times Dom(X_{ik})$, qui spécifie les combinaisons de valeurs (tuples) autorisées pour les variables de Var(C_i). $|Var(C_i)|$ est l'arité de la contrainte C_i.*

Notons que les contraintes peuvent être exprimées en énumérant les tuples autorisés (représentation en extension), ou bien en utilisant des contraintes globales déjà définies (représentation en intention) comme par exemple la contrainte « Alldifferent » qui ne permet pas à ces variables de prendre les mêmes valeurs ou les contraintes de cardinalités qui éliminent les tuples avec une somme supérieure à un seuil donné.

Exemple 1.14. *Reprenons l'exemple 1.13. Considérons les deux contraintes binaires suivante : C_1 : $X_1+X_2 \leqslant 5$ et $C_2 : X_1 \neq X_2$. La première contrainte C_1 est équivalente à $\{(0, 1), (0, 3), (0, 5), (1, 1), (1, 3)\}$ tandis que C_2 est équivalente à $\{(0, 1), (0, 3), (0, 5), (1, 3), (1, 5))\}$.*

Définition 1.20 (CSP). *Un prblème de satisfaction de contraintes (CSP), est un triplet $\langle \mathcal{V}, D, C \rangle$ tels que :*

- *$\mathcal{V}$ est un ensemble de variables ;*
- *D est l'ensemble des domaines de variables de $\mathcal{V}$;*
- *C est un ensemble de contraintes définies sur des sous-ensembles de $\mathcal{V}$.*

Définition 1.21. *Soit $P = \langle \mathcal{V}, D, C \rangle$ un CSP avec $\mathcal{V} = \{X_1, \ldots, X_n\}$, $D = \{Dom(X_1), \ldots, Dom(X_n)\}$ et $C = \{C_1, \ldots, C_k\}$. Un tuple $t = (v_1, \ldots, v_n) \in Dom(X_1) \times \ldots \times Dom(X_n)$ est une solution de P si et seulement si t satisfait toutes les contraintes dans C.*

Exemple 1.15. *Soit le CSP P correspondant à l'exemple 1.14.*
variables : $X_1 \in \{0, 1\}$, $X_2 \in \{1, 3, 5\}$;
contraintes : $X_1 \neq X_2$, $X_1 + X_2 \leqslant 5$;
Le tuple (0,1) est une solution de P car $0 \neq 1$ et $0 + 1 = 1 \leqslant 5$. Par contre le tuple (1,1) n'est pas une solution car la première contrainte est violée.

Un problème d'optimisation sous contraintes (COP) est un CSP $P = \langle \mathcal{V}, D, C \rangle$ avec une fonction objectif f à optimiser tel que $f : Dom(X_1) \times \ldots \times Dom(X_n) \rightarrow \mathbb{R}$. Une solution de COP est une solution de P qui optimise la fonction f.

Exemple 1.16. *Reprenons le CSP de l'exemple 1.15 et considérons la fonction objectif maximiser X_1+X_2. Le tuple (0,5) est la seule solution optimale de ce COP.*

En général, décider la satisfiablité d'un CSP est un problème NP-complet et trouver la solution d'un COP est un problème NP-difficile. Cependant, les techniques utilisées dans les solveurs CP modernes permettent de résoudre efficacement de nombreux instances difficiles. Pour résoudre un problème réel en utilisant CP, le problème doit être modélisé comme un CSP ou un COP en précisant :

— les variables avec leurs domaines ;
— les contraintes reliant les variables ;
— la fonction objectif dans le cas de COP.

Il existe plusieurs solveurs efficaces pour résoudre les CSPs et les COPs et on peut citer comme exemples Gecode [STL10] et Choco [JRL08]. Pour plus de détails sur la programmation par contraintes voir [Lec13].

1.4 Conclusion

Dans ce chapitre, nous avons présenté la satisfiabilité propositionnelle et la programmation par contrainte. Pour bien comprendre le problème SAT, nous avons commencé par la présentation de la logique propositionnelle. Puis, nous avons détaillé comment transformer une formule propositionnelle quelconque vers une forme normale conjonctive. Ensuite, le problème SAT a été défini et plusieurs algorithmes qui permettent sa résolution ont été décrits. Pour finir la présentation de la satisfiabilité, une description de quelques encodages de la fameuse contrainte de cardinalité avec un bref aperçu sur quelques problèmes autour de SAT ont été fournis. Pour finir ce chapitre, nous avons présenté les éléments nécessaire pour modéliser un problème donné en CP. SAT et CP sont utilisées dans la suite de cette thèse pour modéliser différents problèmes de clustering et de fouille de règles d'association.

Chapitre 2

Clustering

Sommaire

Une des plus importantes approches pour analyser les données est de les regrouper dans des catégories ou des classes, de telle sorte que les objets qui appartiennent au même groupe ont des propriétés similaires en se basant sur des critères bien définis. Cette approche s'appelle classification. On distingue deux types de classification : on parle de classification supervisée si on connaît a priori les descriptions des classes qu'on veut, sinon si les classes sont construites à partir des caractéristiques des objets alors on parle de classification non supervisée ou clustering.

Le clustering est une technique visant à partitionner un ensemble d'objets en sous-ensembles homogènes et bien séparés appelés clusters. L'homogénéité signifie que les objets dans un même cluster sont très similaires par contre la séparation exprime qu'un objet dans un cluster doit être différent des objets des autres clusters.

Le problème de clustering a été largement étudié pour son application dans plusieurs domaines à savoir la biologie [WEL+10], l'analyse des images [MK95], et dans le commerce [WXL99], etc. En effet plusieurs algorithmes ont été proposés pour différents types de données, citant comme exemples

les algorithmes basés sur une mesure de distance entre les objets comme le k-$means$ [Mac67], les algorithmes hiérarchiques [SS62, KHK99] et les algorithmes basés sur la densité [EKSX96].

Dans ce chapitre nous présentons un bref aperçu de quelques méthodes de clustering. Nous nous concentrons davantage sur les méthodes basées sur une mesure de distance qui sont plus proches de nos contributions. Nous présentons également une vue d'ensemble du problème de clustering sous contraintes.

2.1 Mesures de distance et de similarité entre les données

Les clusters sont considérés comme des groupes qui contiennent des objets similaires entre eux et différents des objets qui appartiennent à d'autres clusters. Naturellement, on se pose les questions suivantes : sur quelles bases cette similarité est déterminée ? ou bien comment mesurer la distance (dissimilarité) ou plus généralement la proximité entre deux objets, entre un objet et un cluster et entre deux clusters. Le choix d'une mesure de similarité dans le clustering est très important et dépend fortement au type de données. Nous présentons dans cette section les mesures les plus utilisées et les plus connues dans la littérature pour chaque type de données.

Définition 2.1 (Distance métrique [XW08])**.** *Soit $\mathcal{O}$ un ensemble d'objets. Une fonction d est dite distance métrique sur $\mathcal{O}$ si et seulement si elle satisfait les conditions suivantes :*

1. *Symétrie :* $\forall o, o' \in \mathcal{O},\ d(o, o') = d(o', o)$;
2. *Positivité :* $\forall o, o' \in \mathcal{O},\ d(o, o') \geqslant 0$;
3. *Réflexivité :* $\forall o \in \mathcal{O},\ d(o, o) = 0$;
4. *Inégalité triangulaire :* $\forall o, o', o'' \in \mathcal{O},\ d(o, o') \leqslant d(o, o'') + d(o'', o')$.

Notons qu'une mesure de dissimilarité doit satisfaire les trois premières conditions et pas forcément l'inégalité triangulaire [XW08]. Si toutes les conditions de 1 à 4 sont satisfaites alors la mesure est appelée *métrique*. La mesure de similarité est définie de la même façon [XW08].

Définition 2.2 (Similarité métrique)**.** *Soit $\mathcal{O}$ un ensemble d'objets. Une fonction s est dite similarité métrique sur $\mathcal{O}$ si et seulement si elle satisfait les conditions suivantes :*

1. *Symétrie :* $\forall o, o' \in \mathcal{O},\ s(o, o') = s(o', o)$;
2. *Positivité :* $\forall o, o' \in \mathcal{O},\ 0 \leqslant s(o, o') \leqslant 1$;
3. *Réflexivité :* $\forall o \in \mathcal{O},\ s(o, o) = 1$;
4. $\forall o, o', o'' \in \mathcal{O},\ s(o, o')s(o', o'') \leqslant [s(o, o') + s(o', o'')]s(o, o'')$.

Notons aussi qu'une mesure de similarité doit satisfaire les trois premières conditions et pas forcément la dernière.

Une grande valeur implique une grande proximité pour une fonction de similarité. Par contre, une petite valeur implique une grande proximité pour une mesure de distance. Dans certains domaines où les données sont quantitatives, on parle généralement de distance tandis que dans d'autres domaines où les données sont plutôt catégorielles, on parle de similarité. Les mesures de distances et de similarité sont très sensibles à la distribution, la dimensionnalité et au type de données. En effet, nous présentons dans ce qui suit quelques mesures de proximité pour divers types de données.

2.1.1 Données quantitatives

L_p-norme est l'une des mesures de distance les plus connues pour les données quantitatives. Elle est définie entre deux points (objets quantitatives) à n dimensions $X = (x_1, \ldots, x_n)$ et $Y = (x_1, \ldots, x_n)$ comme suit :

$$d(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2.1}$$

Notons que la distance euclidienne (L_2) est un cas particulier de L_p-norme avec $p = 2$. Elle est définie par l'équation suivante :

$$d(X,Y) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2} \tag{2.2}$$

La distance euclidienne est couramment utilisée pour évaluer la proximité entre les objets quantitatifs surtout pour des espaces à deux ou trois dimensions [Mac67]. Elle donne des clusters de formes hyper-sphériques. De plus, cette distance a une propriété d'invariance par rapport aux rotations, c-à-d, les clusters formés en utilisant la distance euclidienne sont invariants par rapport aux translations et rotations dans l'espace des caractéristiques, car une ligne droite entre deux point ne change pas avec le changement des axes [Agg15].

En utilisant la distance euclidienne, la classification donne de bons résultats lorsque l'ensemble des données contient des clusters qui sont compacts ou isolés [JMF99]. Cependant, si les caractéristiques (attributs) sont mesurées sur des échelles largement différentes, alors les attributs avec de grandes valeurs et variances vont dominer les autres. Une des solutions possibles à ce problème est de normaliser les données de telles sorte que toutes les caractéristiques contribuent équitablement au calcul de la distance. Il existe plusieurs normalisations possibles qui ont été détaillées dans [XW08].

Un autre cas particulier de L_p-norme est la distance de Manhattan (L_1) avec $p = 1$. Cette distance est intuitivement dérivée du fait que dans une ville où les rues sont représentées par une grille comme le cas de l'arrondissement de Manhattan à New York, la distance euclidienne ne peut être appliquée. En effet, la distance de Manhattan est définie comme suit :

$$d(X,Y) = |x_1 - y_1| + \ldots + |x_n - y_n| \tag{2.3}$$

Exemple 2.1. *Soit $X = (1,2,3)$ et $Y = (2,4,6)$ deux points dans un espace continu à trois dimensions. La distance Euclidienne $d_e(X,Y) = \sqrt{(1-2)^2 + (2-4)^2 + (3-6)^2} = \sqrt{14} = 3,74$. Par contre la distance de Manhattan est $d_m(X,Y) = |1-2| + |2-4| + |3-6| = 6$.*

Les fonctions de distance L_p sont très sensibles au bruit, au nombre de dimensions et à la distribution des données. En effet des améliorations et corrections ont été proposées. Pour plus de détails voir [Agg15].

2.1.2 Données catégorielles

Pour les données catégorielles, les mesures de similarité sont les plus adéquates et les plus couramment utilisées par rapport aux distances. Considérons d'abord le cas des objets avec des variables (caractéristiques ou attributs) binaires (domaine avec deux valeurs possibles).

Les caractéristiques binaires peuvent être classifiées en deux catégories, symétrique ou asymétrique [XW08]. Cette classification est basée sur le fait que les deux valeurs (0 et 1) ont la même importance ou non. Dans une caractéristique symétrique, les deux valeurs ont le même poids. Par contre dans une caractéristique asymétrique une valeur (généralement 1) a un poids plus important que l'autre [XW08].

La mesure de similarité symétrique regarde les appariements de 1-1 et 0-0 et elle les traite équitablement comme le montre la mesure suivante :

$$s(X,Y) = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + w(n_{10} + n_{01})} \tag{2.4}$$

Où n_{ij} représente le nombre de variables qui ont la valeur i dans X et la valeur j dans Y. Notons que pour le cas de $w = 1$, cette mesure de similarité représente le coefficient d'appariement simple [Zub38] et la mesure de dissimilarité $d(X,Y) = 1 - s(X,Y)$, représente la distance de Hamming [Ham50]. Pour $w = 2$ nous obtenons la mesure de Rogers et Tanimoto [RT60] où les non-appariements comptent double. Enfin, pour $w = \frac{1}{2}$ nous obtenons la mesure de Sokal et Sneath [Sok63] où les appariements comptent double.

La mesure de similarité suivante se concentre sur l'appariement 1-1 des caractéristiques et ignore l'effet d'appariement 0-0 car elle considère que la valeur 1 est plus significative que la valeur 0.

$$s(X,Y) = \frac{n_{11}}{n_{11} + w(n_{10} + n_{01})} \tag{2.5}$$

Notons que pour le cas de $w = 1$, la mesure de similarité représente le coefficient de Jaccard [Jac12] et pour $w = \frac{1}{2}$ nous obtenons l'indice de Dice [Dic45].

Considérons maintenant le cas des objets avec des variables non binaires. Dans ce cas, la première solution est de transformer les variables non binaires en plusieurs variables binaires [KR09]. Par exemple, si on a une caractéristique qui représente la taille d'une personne et qu'elle peut prendre une des trois valeurs grand, moyen et petit, alors pour chaque valeur une variable binaire est créée telle que la variable qui prend la valeur 1 indique son occurrence. Par exemple, si la taille d'une personne est petite, alors la variable x_{petit} aura la valeur 1 et les deux variables x_{grand} et x_{moyen} auront la valeur 0.

Une autre méthode basée sur un critère d'appariement simple est couramment utilisée, telle que pour une paire d'objets $X = \{x_1, \ldots, x_n\}$ et $Y = \{y_1, \ldots, y_n\}$, la similarité est obtenue comme suit :

$$s(X,Y) = \frac{1}{n}\sum_{i=1}^{n} s_i \tag{2.6}$$

Où $s_i = w$ s'il y a un appariement entre X et Y pour la caractéristique i et $s_i = 0$ sinon. La valeur w peut être une valeur fixe pour toutes les caractéristiques (généralement 1) ou une valeur spécifique a chacune.

2.1.3 Données quantitatives et catégorielles (mixtes)

L'idée principale pour mesurer la similarité entre les données qui contiennent à la fois des variables catégorielles et des variables quantitatives est d'utiliser deux mesures de similarités pour les deux types de variables. Considérons les deux objets $X = (X_c, X_q)$ et $Y = (Y_c, Y_n)$ où X_c, Y_c sont les sous-ensembles avec des attributs catégoriels et X_q, Y_q sont les sous-ensembles d'attributs quantitatifs. La

mesure de similarité globale entre X et Y est alors définie comme suit [Agg15] :

$$s(X,Y) = \lambda\ s_q(X_q, Y_q) + (1-\lambda)\ s_c(X_c, Y_c) \tag{2.7}$$

Où s_q (resp. s_c) est une mesure de similarité pour des données quantitatives (resp. catégorielles) et $0 < \lambda < 1$. Le grand challenge pour cette mesure est de décider les poids des composantes catégorielles et quantitatives ainsi que la normalisation. Pour plus de détails voir [Agg15].

2.1.4 Données textuelles

Les données textuelles peuvent être considérées comme des données quantitatives multidimensionnelles où chaque mot représente un attribut et la fréquence du mot dans le texte est sa quantité. Cependant, on ne peut pas utiliser les mesures telles que L_p-norme car la structure de texte est très éparpillée et la plupart des attributs auront une valeur 0. De plus, toutes les fréquences sont positives. En effet, en utilisant par exemple la distance euclidienne sur deux textes long, le résultat est en général plus grand qu'entre deux petits textes bien que les deux textes longs partagent plus de mots que les deux autres. Pour résoudre ce problème la mesure cosine est la plus utilisée [LA99]. Elle calcule l'angle entre deux documents ce qui n'est pas sensible à la longueur du texte. Elle est définie entres deux objets textuels $X = \{x_1, \ldots, x_n\}, Y = \{y_1, \ldots, y_n\}$ par l'équation suivante :

$$cosine(X,Y) = \frac{\sum_{i=1}^{n} x_i.y_i}{\sqrt{\sum_{i=1}^{n} {x_i}^2}.\sqrt{\sum_{i=1}^{n} {y_i}^2}} \tag{2.8}$$

2.1.5 Données sous formes de graphes

La similarité pour les graphes peut être mesurée de différentes façons selon qu'elle est calculée entre deux graphes ou entre deux nœuds d'un même graphe.

— *Similarité entre deux nœuds d'un même graphe* : Deux nœuds d'un graphe sont très proches l'un de l'autre s'ils sont connectés via des arêtes. Par conséquent, les nœuds qui sont connectés par des chemins courts et plusieurs chemins doivent être considérés comme proches. La similarité basée sur la structure mesure la proximité entre deux nœuds par le plus court chemin. Par contre elle ne prend pas en compte la multiplicité des chemins. En effet, la mesure de similarité basée sur le chemin aléatoire qui prend en compte les visites multiples d'un même nœud pour calculer la similarité. Pour plus de détails voir [FPRS07].

— *Similarité entre deux graphes* : Mesurer la similarité entre deux graphes est un challenge car le simple problème de décider si deux graphes sont identiques est très dur (problème d'isomorphisme). Plusieurs mesures de similarité ont été proposées. La première mesure la distance entre deux graphes en calculant le plus grand sous graphes en commun. La deuxième mesure, appelée *graph-edit*, définie la distance entre deux graphes par le nombre de modifications faites pour transformer un graphe vers l'autre [SF83]. Enfin, la mesure basée sur les sous structures compte le nombre de sous structures fréquentes entres les deux graphes [AR13].

2.2 Mesures de qualité d'un clustering

L'objectif d'un algorithme de clustering est de partitionner un ensemble d'objet $O = \{o_1, \ldots, o_n\}$ en k sous-ensembles disjoints de telle sorte que les objets qui se trouvent dans une même partition sont plus proches par rapport aux objets des autres partitions. Ce problème est généralement exprimé comme un problème d'optimisation qui consiste à trouver la partition qui maximise un critère de qualité.

2.2.1 Qualité d'un cluster

La qualité d'un cluster est généralement représentée par des mesures d'homogénéité et de séparation. En effet, de nombreux critères ont été proposés pour mesurer l'homogénéité d'un cluster et nous pouvons citer comme exemples [HJ97] :

— *Diamètre* : Le diamètre d'un cluster $\mathcal{C}$ est la distance entre les deux objets les plus éloignés et il est calculé comme suit :

$$diameter(\mathcal{C}) = \max_{o,o' \in \mathcal{C}} d(o, o') \tag{2.9}$$

— *Rayon* : Le rayon d'un cluster $\mathcal{C}$ est la plus petite distance pour tous les objets o parmi les plus grandes distances entres toute paire (o, o') :

$$radius(\mathcal{C}) = \min_{o \in \mathcal{C}} \max_{o' \in \mathcal{C}} d(o, o') \tag{2.10}$$

— *Étoile* : Cette mesure calcule le minimum entre les sommes de distances entre un objet $o \in \mathcal{C}$ et le reste des objets de $\mathcal{C}$:

$$star(\mathcal{C}) = \min_{o \in \mathcal{C}} \frac{1}{|\mathcal{C}| - 1} \sum_{o' \in \mathcal{C}} d(o, o') \tag{2.11}$$

— *Clique* : La clique est la somme des distances entre toute paire d'objets o, o' dans un cluster $\mathcal{C}$:

$$clique(\mathcal{C}) = \frac{1}{|\mathcal{C}|(|\mathcal{C}| - 1)} \sum_{o,o' \in \mathcal{C}} d(o, o') \tag{2.12}$$

— *Somme des carrés* : Soit $\mathcal{C}$ un cluster d'objets dans un espace euclidien et $\overline{o}$ le centre du cluster. Cette mesure est égale à la somme des carrés des distances euclidiennes entre chaque objet $o \in \mathcal{C}$ et le centre $\overline{o}$:

$$ss(\mathcal{C}) = \sum_{o_i \in \mathcal{C}} (||o_i - \overline{o}||_2)^2 \tag{2.13}$$

Où $||.||_2$ représente la distance euclidienne et chaque attribut i de $\overline{o}$ est calculé comme suit :

$$\overline{o}_i = \frac{1}{|\mathcal{C}|} \sum_{o \in \mathcal{C}} o_i \tag{2.14}$$

o_i représente l'attribut i de l'objet o.

— *Variance* : Elle est égale à la somme des carrés divisée par la taille du cluster :

$$var(\mathcal{C}) = \frac{ss(\mathcal{C})}{|\mathcal{C}|} \tag{2.15}$$

La séparation d'un cluster $\mathcal{C}$ est mesurée en utilisant les critères suivants [HJ97] :

— *Écart « split »* : Ce critère calcule le minimum des distances entre tout objet o d'un cluster $\mathcal{C}$ et le reste des objets qui n'appartiennent pas à ce dernier :

$$split(\mathcal{C}) = \min_{o \in \mathcal{C}, o' \notin \mathcal{C}} d(o, o') \tag{2.16}$$

— Coupure « cut » : Cette mesure calcule la somme des distances entre les objets d'un cluster $\mathcal{C}$ et le reste des objets qui n'appartiennent pas à $\mathcal{C}$:

$$cut(\mathcal{C}) = \sum_{o \in \mathcal{C}} \sum_{o' \notin \mathcal{C}} d(o, o') \tag{2.17}$$

2.2.2 Qualité d'une partition

La qualité d'une partition $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ est également représentée par des mesures d'homogénéité et de séparation. Ces deux critères sont des agrégations des mesures d'homogénéité et de séparation des différents clusters. L'homogénéité de la partition P est donnée soit par le maximum des homogénéités ou par la somme des homogénéités des clusters :

— *Diamètre* d'une partition :

$$diameterP(P) = \max_{1 \leqslant i \leqslant k} diameter(\mathcal{C}_i) \tag{2.18}$$

— *Somme des diamètres* d'une partition :

$$SdiameterP(P) = \sum_{1 \leqslant i \leqslant k} diameter(\mathcal{C}_i) \tag{2.19}$$

— *Somme des cliques* d'une partition :

$$cliqueP(P) = \sum_{1 \leqslant i \leqslant k} clique(\mathcal{C}_i) \tag{2.20}$$

— *Somme des erreurs quadratiques* d'une partition :

$$ssP(P) = \sum_{1 \leqslant i \leqslant k} ss(\mathcal{C}_i) \tag{2.21}$$

Notons que la plupart de ces critères sont NP-difficiles à l'exception de la mesure d'écart $splitP$.

La séparation d'une partition est généralement donnée par une des mesures suivantes :

— *Écart d'une partition « split »* :

$$splitP(P) = \min_{1 \leqslant i \leqslant k} split(\mathcal{C}_i) \tag{2.22}$$

— *Somme des écarts* d'une partition :

$$SsplitP(P) = \sum_{1 \leqslant i \leqslant k} split(\mathcal{C}_i) \tag{2.23}$$

— *Coupure « ratio cut »* est mesuré par [HK92] :

$$ratioCut(P) = \frac{1}{2} \sum_{1 \leqslant i \leqslant k} \frac{cut(\mathcal{C}_i)}{|\mathcal{C}_i|} \tag{2.24}$$

Il est à noter qu'une grande séparation signifie un bon clustering.

2.3 Algorithmes pour le clustering

Le problème de clustering a été largement étudié ces dernières années par rapport a son application dans différents domaines et surtout dans la fouille de données. En effet, plusieurs approches et algorithmes ont été proposés, citant comme exemples, les méthodes hiérarchiques, les méthodes de partitionnement et les approches basées sur la densité [Agg15].

Dans cette section nous présentons quelques unes de ces approches que nous avons jugées intéressantes et proches de nos contributions. Nous montrons les avantages et les inconvénients de chacune que ce soit en temps de calcul ou en terme de qualité des clusters fournies. Avant de présenter ces méthodes de clustering, donnons quelques définitions nécessaires par la suite.

Définition 2.3 (Partition)**.** *Soit $\mathcal{O}$ un ensemble d'objets et $\mathcal{C}_1, \ldots, \mathcal{C}_k$ des sous-ensemble de $\mathcal{O}$. L'ensemble $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ est dite partition de $\mathcal{O}$ si et seulement si :*

— $\forall i \in \{1, \ldots, k\},\ \mathcal{C}_i \neq \emptyset$;
— $\forall i, j \in \{1, \ldots, k\}, i \neq j,\ \mathcal{C}_i \cap \mathcal{C}_j = \emptyset$
— $\forall i \in \{1, \ldots, k\},\ \bigcup \mathcal{C}_i = \mathcal{O}$;

La plupart des algorithmes de clustering choisissent un représentant d'un cluster pour simplifier les calculs des distances et des qualités. Le centroïde (resp. médoïde) est généralement utilisé pour représenter les objets quantitatifs (resp. quelconques).

Définition 2.4 (Centroïde)**.** *Soit $\mathcal{C}$ un cluster donné. Le centroïde ,noté $\overline{o}$, est défini par :*

$$\overline{o}_i = \frac{1}{|\mathcal{C}|} \sum_{o \in \mathcal{C}_c} o_i \tag{2.25}$$

Notons que o_i représente l'attribut i de l'objet o.

Définition 2.5 (Médoïde)**.** *Soit $\mathcal{C} = \{o_1, \ldots, o_n\}$ un cluster donné. Le médoïde ,noté o_m, est un objet o de $\mathcal{C}$ qui à la plus petite somme de distances avec les autres objets de $\mathcal{C}$:*

$$\underset{o \in \mathcal{C}}{min} \frac{1}{|\mathcal{C}|} \sum_{o' \in \mathcal{C}} d(o, o') \tag{2.26}$$

Notons que o_m est un objet du cluster alors que $\overline{o}$ peut ne pas l'être.

2.3.1 Algorithmes hiérarchiques

Les algorithmes hiérarchiques ont été développés dans le but de construire des mécanismes flexibles et déterministes pour le problème de clustering. Les méthodes hiérarchiques peuvent être divisées en deux grandes catégories, ascendantes « agglomerative » et descendantes « divisive ». L'approche ascendante commence par des clusters singletons (chaque cluster contient un seul objet) puis elle fusionne itérativement les deux clusters les plus proches. Par contre l'approche descendante considère au départ que tous les objets sont dans un seul cluster puis elle divise itérativement un cluster choisi en deux. Le choix des clusters à fusionner ou à diviser est effectué selon un critère donné (voir section 2.2.1). Ces approches construisent un arbre, appelé « dendogramme », qui représente une hiérarchie à plusieurs niveaux décrivant les liens de proximités entre les objets à chaque fois qu'on descend ou qu'on remonte dans cet arbre.

Méthode ascendante : L'algorithme 4 représente les étapes de base d'un algorithme hiérarchique ascendant. Tout d'abord une matrice de proximité entre tous les objets est construite en utilisant une mesure adéquate pour le type d'objets à regrouper (ligne 1). Chaque objet représente un cluster à cette étape (ligne 2). Ensuite, les deux clusters les plus proches dans P sont choisis et fusionnés à chaque itération en regardant la matrice de distance M (ligne 4-5). La matrice de dissimilarité est mise à jour après chaque étape de fusion en remplaçant les clusters fusionnés par leur union (ligne 6). Le processus de fusion des clusters continue jusqu'à obtention d'un seul cluster qui contient tous les objets ou satisfaction d'un critère d'arrêt.

Algorithm 4: Algorithme hiérarchique ascendant

Input: Un ensemble d'objet $\mathcal{O} = \{o_1, \ldots, o_n\}$
Output: Une partition $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$
1 Initialiser M une matrice de distance de taille $n * n$;
2 $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ où $\mathcal{C}_i = \{o_i\}$;
3 **while** $|P| \neq 1$ *et critère* $= faux$ **do**
4 $(\mathcal{C}_i, \mathcal{C}_j)$=Choisir($P$,$M$);
5 Fusionner($\mathcal{C}_i$, $\mathcal{C}_j$, P);
6 MiseAjour(M);
7 **return** P;

Les algorithmes $single - link$ [SS62] et $complete - link$ [Kin67] sont les deux versions de l'approche hiérarchique ascendante les plus connues. La différence entre ces deux méthodes est la mesure de distance entre deux clusters. La distance entre deux cluster $\mathcal{C}_i, \mathcal{C}_j$ pour l'algorithme $single - link$ est celle des plus proches voisins « nearest neighbor » entre leurs membres $d(\mathcal{C}_i, \mathcal{C}_j) = splitP(\{\mathcal{C}_i, \mathcal{C}_j\})$. Cette méthode donne plus d'importance aux régions qui contiennent des objets proches en négligeant les structures globales propres aux clusters. Dans l'approche $complete - link$ [Kin67], la mesure de distance entre deux clusters est celle des deux objets les plus éloignés. Elle est équivalente au choix de la paire de clusters qui donne après la fusion un cluster avec le plus petit diamètre possible. Cette approche prend en considération la structure interne des clusters et permet d'avoir des partitions compactes. Les deux approches $single - link$ et $complete - link$ sont sensibles aux bruits [Agg15]. Néanmoins, elles se comportent bien pour les données bien séparées et concentrées. En effet, l'algorithme $single - link$ fournit une partition optimale par rapport au critère de l'écart « split » à tous les niveau du *dendogramme* [DH80].

Méthode descendante : Comparé au clustering hiérarchique ascendant, la méthode descendante procède dans le sens opposé. Au début, tous les objets sont dans un même cluster et récursivement ce dernier va être divisé jusqu'à atteindre le cas où chaque objet est dans son propre cluster. Cet approche descendante a l'avantage d'être plus efficace surtout dans le cas où le nombre de niveaux (le nombre de clusters) à générer est très petit par rapport au nombre d'objets, ce qui est généralement le cas. De plus cette approche est dite globale car elle contient toute la structure au début, ce qui réduit le cumul des erreurs par rapport aux méthodes ascendantes.

L'algorithme 5 décrit l'approche descendante de base. D'abord, tous les objets sont dans un même cluster (ligne 1). Puis, on divise itérativement un cluster choisi (ligne 3-5) en un ensemble de clusters (2 ou plus) en utilisant une méthode bien définie (algorithme $\mathcal{A}$), jusqu'à atteindre un critère de coupure ou atteindre le cas de base où chaque cluster est un singleton ou satisfaire un critère d'arrêt. La mise à jour (ligne 6) est effectuée à chaque itération en remplaçant le cluster divisé par le résultat de la division P_1.

Le choix d'un cluster pour l'étape de division s'effectue en utilisant un critère donné comme par exemple la somme des erreurs carrées (Voir section 2.2.1).

Algorithm 5: Algorithme hiérarchique descendant

Input: Un ensemble d'objet $\mathcal{O} = \{o_1, \ldots, o_n\}$;
Un algorithme de division $\mathcal{A}$
Output: Une partition $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$

```
P = {O} ;
while |P| ≠ n et critère = faux do
    C_i=ChoisirCluster(P);
    P_1=Diviser(C_i, A);
    MiseAjour(P,P_1);
return P;
```

Le « bisecting k-$means$ » [SB01] est un algorithme ascendant où chaque cluster choisi pour la division est partitionné en exactement deux sous clusters en utilisant un algorithme 2-$means$ (voir section 2.3.2). Plusieurs autres variantes ont été proposées en changeant généralement, soit le critère de choix pour la division, soit l'algorithme de division. Ces différentes méthodes et critères de division visent l'équilibrage des clusters [AR13].

Les algorithmes hiérarchiques présentés précédemment se comportent bien dans le cas ou les clusters ont des formes géométriques sphériques et des tailles équivalentes. Par contre, ils ne donnent pas de bons résultats dans le cas où les clusters ont des formes non sphériques et ils n'arrivent pas à les capturer. De plus, ils sont très sensibles aux bruits. Pour régler ces problèmes plusieurs algorithmes ont été proposés dans la littérature comme par exemples $CURE$ [GRS98] et $Chameleon$ [KHK99].

2.3.2 Algorithmes basés sur les représentants

Les algorithmes basés sur les représentants sont les plus simples des algorithmes de clustering car ils sont liés directement à la notion de distance et de similarité entre les objets. Les clusters dans ce type d'algorithmes sont créés d'un seul coût et il n'existe pas de relation hiérarchique entre différents clusters. L'idée principale de ces méthodes est de trouver un ensemble de représentants de bon qualité pour trouver de bons clusters. Une fois les représentant sont déterminés, une simple mesure de distance peut être utilisée pour affecter les objets au plus proche représentant.

Étant donné un nombre de clusters k et un ensemble d'objets $\mathcal{O}$, le but est de déterminer un ensemble de représentants $R = \{\overline{o}_1, \ldots, \overline{o}_k\}$ qui minimise la fonction objectif suivante :

$$\sum (\min_{\overline{o} \in R} \; d(o, \overline{o})) \quad \forall o \in \mathcal{O} \tag{2.27}$$

En d'autres termes, la somme des distances entre les différents objets et leurs plus proche représentants doit être minimisée.

Présentons maintenant les plus connus des algorithmes de partitionnement basés sur les représentants.

k-$means$: L'algorithme k-$means$ (k-moyennes), proposé dans [Mac67], est la version basée sur les représentants la plus simple et la plus utilisée dans le clustering de données quantitatives. Il commence par choisir k objets comme centroïdes (ou représentants) initiaux pour représenter les k partitions que nous voulons avoir en sortie. Ensuite, chaque objet est affecté à la même partition que celle du plus proche centroïde, en utilisant une mesure de proximité choisie. Une fois tous les objets affectés aux clusters, une mise à jour des centroïdes est faite (voir définition 2.4). Tant que les centroïdes changent, l'algorithme répète itérativement les deux étapes précédentes. Cette approche est décrite dans l'algorithme 6.

L'algorithme 6 commence par choisir aléatoirement k objets comme centroïdes initiaux (ligne 1). Ensuite, les deux étapes d'affectation de chaque objet au cluster le plus proche (ligne 3-5) et de mise à jour des centroïdes sont effectuées jusqu'à stabilisation des centroïdes. L'algorithme k-$means$ minimise la somme des erreurs quadratiques (voir ssP dans la section 2.2.2). Minimiser la fonction ssP est un problème NP-difficile. Néanmoins, le k-$means$ est une heuristique qui garantit la convergence vers un minimum local.

Algorithm 6: Algorithme k-$means$

Input: Un ensemble d'objet $\mathcal{O} = \{o_1, \ldots, o_n\}$ et un nombre de clusters k
Output: Une partition $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$

1 $R = (\overline{o}_1, \ldots, \overline{o}_k) =$ sélectionnerAleatoirement($\mathcal{O}$, k);
2 **while** R *change* **do**
  /* affecter les objets au plus proche centroide */
3 **for** $i = 1$ *à* n **do**
4 $j = \underset{j \in \{1,\ldots,k\}}{\arg\min} (d(o_i, \overline{o}_j))$;
5 $C_j = C_j \cup \{o_i\}$;
  /* mettre à jour les centroides */
6 **for** $i = 1$ *à* k **do**
7 miseAjour($\overline{o}_i, \mathcal{C}_i$);
8 **return** P;

Les facteurs majeurs qui peuvent influer sur la performance du k-$means$ sont le choix des centroïdes initiaux et l'estimation du nombre de clusters k. Plusieurs contributions ont été proposées pour améliorer le choix initial des représentants. On peut citer comme exemple la proposition de Harting et Wong dans [HW79] qui suggère que les objets bien séparés et qui ont un grand nombre d'objets dans leurs proches voisinages sont de bons candidats pour être des représentants initiaux. Dans l'algorithme k-$means$++ [AV07] le premier représentant est choisi aléatoirement puis le prochain est le plus éloigné du premier. Cette sélection est basée sur une fonction de probabilité. La sélection continue jusqu'à avoir les k représentants initiaux. Récemment dans [CKV13], une étude comparative entre plusieurs méthodes d'initialisation a été faite et des recommandations ont été proposées après une analyse des résultats expérimentaux. Un autre inconvénient du k-$means$ est la spécification du nombre de clusters k en entrée. Plusieurs propositions ont été présentées permettant de donner une bonne estimation du nombre de partitions k en optimisant des fonctions objectifs avec k comme paramètre. On peut citer, l'index de Calinski-Harbasz [CH74] et le critère d'information d'Akaik (AIC) [YFM+01].

k-$medoids$: L'algorithme k-$means$ est appliqué uniquement pour les données quantitatives. De plus, il est très sensible au bruit car les objets représentant ce dernier sont utilisés dans le calcul des moyennes.

En effet, l'algorithme $k\text{-}medoids$ est très résilient au bruit [AR13]. Il utilise un objet réel pour représenter un cluster contrairement au $k\text{-}means$ qui utilise un objet fictif calculé par une moyenne. L'idée de base est décrite dans l'algorithme 7. Il commence par un choix aléatoire des représentants parmi les objets. Puis, chaque objet et assigné au plus proche représentant. Ensuite, l'algorithme choisi aléatoirement un objet pour remplacer un des représentant si le coût de changement est positif. Enfin, une mise à jour de la partition et effectuée. Ces deux dernières étapes de choix et de remplacement sont répétées jusqu'à satisfaction d'un critère d'arrêt ou aucun changement ne peut être effectué.

Algorithm 7: Algorithme k-medoids

Input: Un ensemble d'objet $\mathcal{O} = \{o_1, \ldots, o_n\}$ et un nombre de clusters k
Output: Une partition $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$

1 $R = (o_{m1}, \ldots, o_{mk}) =$ sélectionnerAléatoirement($\mathcal{O}$, k);
2 **while** *critère* = $faux$ **do**
  /* affecter les objets au plus proche représentant */
3 **for** $i = 1$ *à* n **do**
4 $j = \underset{j \in \{1,\ldots,k\}}{\arg\min} (d(o_i, o_{mj}))$;
5 $C_j = C_j \cup \{o_i\}$;
  /* mettre à jour les représentant */
6 ChoisirAléatoirement($o_c \notin R$);
7 S=coûtDeChangement(o_c, $o_m \in R$);
8 **if** $S>0$ **then**
9 changer(o_c, o_m,R);
10 miseAjour(P)
11 **return** P;

D'autres versions du $k\text{-}means$ ont été proposées comme le $k\text{-}medians$ qui calcule la médiane pour représenter un cluster au lieu d'utiliser la moyenne. Le $k\text{-}modes$ est une version proposée pour partitionner les données catégorielles [Hua98]. Pour plus de détails voir [AR13].

L'objet le plus éloigné en premier : L'approche FPF « Furthest Point First » a été introduite dans [Gon85]. Elle a comme objectif d'optimiser un critère de diamètre (voir section 2.2.2). Dans cette méthode, chaque cluster a un objet qui le représente. Chaque objet dans un cluster est plus proche au représentant de son cluster qu'aux autres représentants des autres clusters. L'algorithme 8 présente les étapes de cette approche. Il commence par choisir un objet comme représentant du premier cluster et affecte tous les objets à ce cluster (ligne 1-2). Ensuite, l'objet le plus éloigné du premier représentant est choisi pour représenter le deuxième cluster. Les objets qui sont plus proches du deuxième représentant qu'au premier sont déplacés vers le deuxième cluster. Ces deux étapes de choix de l'objet le plus éloigné du représentant de son cluster (ligne 5) et du déplacement des objets qui sont plus proches au nouveau représentant (ligne 6) sont répétés jusqu'à obtention de k clusters.

2.3.3 Algorithmes basés sur la densité

La plupart des algorithmes basés sur les points représentatifs comme le $k\text{-}means$ sont appropriés pour extraire des clusters convexes tels que les clusters de formes sphériques ou ellipsoïdales. Cependant, les données peuvent prendre d'autres formes géométriques à cause des contraintes imposées par la nature

Algorithm 8: Algorithme FPF

Input: Un ensemble d'objet $\mathcal{O} = \{o_1, \ldots, o_n\}$ et un nombre de clusters k
Output: Une partition $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$

$o_{r1} =$ ChoisirAléatoirement($\mathcal{O}$);
$\mathcal{C}_1 = \{\mathcal{O}\}$;
$P = \{\mathcal{C}_1\}$;
$i = 2$;
while $i < k$ **do**
  $o_{ri} = \underset{o \in \mathcal{C}_j}{argmax}\ d(o, o_{rj}) \ \ 1 \leqslant j < i$;
  $\mathcal{C}_i = \{o \in \mathcal{C}_j \ \forall j < i,\ d(o, o_{ri}) < d(o, o_{rj})\}$;
  $P = P \cup \mathcal{C}_i$;
  $i++$
return P;

comme les montagnes, les rivières, etc. En effet, les méthodes de clustering basées sur la densité ont été proposées pour :

— détecter des formes arbitraires des clusters ;
— détecter et supprimer le bruit et les objets aberrants ;
— traiter efficacement les ensembles de données de grandes tailles.

Nous présentons maintenant les plus connus des algorithmes basés sur la densité tout en considérant que les objets sont des vecteurs numériques, appelés points.

$DBSCAN$ **« Density-Based Spatial Clustering of Applications with Noise » :** Avant de présenter l'algorithme, nous commençons par définir quelques notions nécessaires pour comprendre le fonctionnement de l'algorithme $DBSCAN$ [EKSX96]. La densité dans un point est estimée par $DBSCAN$ en comptant le nombre de points dans un voisinage de rayon fixé, noté Eps. Cette mesure de densité permet de classifier les points en trois catégories par rapport à un seuil $MinPts$ fixé par l'utilisateur :

1. *point cœur* : Un point est défini comme un point cœur s'il a au minimum $MinPts$ points dans son voisinage de rayon Eps.
2. *point frontière* : Un point est défini comme un point frontière s'il a moins de $MinPts$ points et au moins un point cœur dans son voisinage de rayon Eps.
3. *point bruit* : C'est un point qui est ni point cœur ni point frontière.

Exemple 2.2. *La figure 2.1 illustre un point cœur A, un point frontière B et un point bruit C pour un rayon donné et un* $MinPts = 10$. *Le point A est un cœur parce qu'il a 10 point dans son voisinage. B est un point frontière parce qu'il a A dans son voisinage et il a une densité inférieur à 10. Enfin, le point C est un bruit car il a une densité de 4 points et il n'a aucun point cœur dans sont voisinage.*

L'algorithme 9 décrit les étapes de bases de *DBSCAN*. Après classification des points en trois catégories points cœurs, points frontières et points bruits (ligne 1), un graphe $\mathcal{G}$ est créé dans lequel les nœuds représentent les points cœurs. Deux noeuds sont connectés si et seulement si les deux points correspondants ont une distance inférieur à Eps (ligne 2). Les composantes connexes de $\mathcal{G}$ sont identifiés pour représenter les clusters des points cœurs (ligne 3). Ensuite, les nœuds frontières sont ajoutés au cluster avec qui ils ont une connectivité élevée (ligne 4). La partition résultante représente la sortie de *DBSCAN*

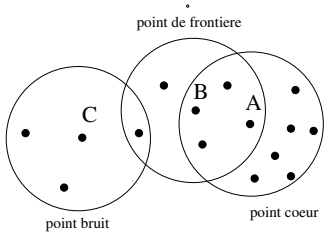


FIGURE 2.1 – Exemples de points cœur, frontière et bruit

et les points bruits sont signalés comme données aberrantes.

Algorithm 9: Algorithme *DBSCAN*

Input: Un ensemble de points $\mathcal{O} = \{o_1, \ldots, o_n\}$, un rayon Eps et un seuil de densité $MinPts$
Output: Une partition P

```
/* Déterminer la catégorie de chaque point : cœur, frontière ou
   bruit                                                        */
```
1 ClassifierPoints($\mathcal{O}$, Eps, $MinPts$);
2 CréerGraphe($\mathcal{G}$, $\mathcal{O}$) ; `/* les points cœurs sont connectés */`
3 P= ComposantesConnexes($\mathcal{G}$);
4 AffecterPointsFrontiere($\mathcal{O}$,P);
5 **return** P;

Notons que *DBSCAN* arrive à détecter des clusters de formes arbitraires et ne prends pas comme paramètre le nombre de cluster k. La complexité de DBSCAN en terme de temps de calcul est de $\mathcal{O}(n^2)$ dans le pire des cas pour un ensemble de n objets.

Clustering basé sur une grille : Dans cette approche [AGGR98], les données sont discrétisées en p intervalles avec une même largeur. Pour des données à d dimensions, la discrétisation génère p^d hyper-cubes. La densité d'une région de la grille est mesurée par le nombre de points qu'elle contient. Un seuil $MinPts$ est utilisé pour déterminer les sous-ensemble des p^d hyper-cubes qui sont denses. Deux régions de la grille sont dites connectées par adjacence s'il partage au moins un coin (ou r surfaces dans le cas général avec $r < d$). En effet, des clusters de formes arbitraires sont formés par les régions denses et connectées de la grille. Deux régions de la grilles sont connectées par densité s'il existe une séquence de régions connectées par adjacence qui les relie. L'objectif d'un clustering basé sur une grille est de déterminer les clusters formés par les régions connectées de la grille.

L'algorithme 10, présente les étapes de base d'une approche basée sur la discrétisation en grille. Il commence par discrétiser l'espace des données en $|R|$ régions (hyper-cubes) avec $|R| = p^d$ (ligne 1). Ensuite, un calcul de densité de chaque région de R est effectué et les régions avec des densités supérieurs à un seuil donné $MinPts$ sont désignées (ligne 2). Pour extraire les clusters de formes arbitraires, un graphe d'adjacence $\mathcal{G}$ est créé où les nœuds représentent les régions et les arêtes sont ajoutés entres les régions denses et adjacentes (ligne 4). Enfin, les clusters sont générés à partir des composantes connexes du graphe $\mathcal{G}$ où chaque composante connexe correspond à un cluster.

Les méthodes basées sur la densité arrivent à extraire des clusters de formes arbitraires. De plus,

Algorithm 10: Algorithme basé sur la discrétisation en grille

Input: Un ensemble de points $\mathcal{O} = \{o_1, \ldots, o_n\}$, nombre d'intervalles p et un seuil de densité $MinPts$

Output: Une partition P

1 R=Discrétiser($\mathcal{O}$, p) ; /* R est un ensemble de régions */
2 RégionsDenses(R, $MinPts$);
3 CréerGraphe($\mathcal{G}$, R) ; /* Deux régions denses sont connectées si elles sont adjacentes */
4 P= ComposantesConnexes($\mathcal{G}$);
5 **return** P;

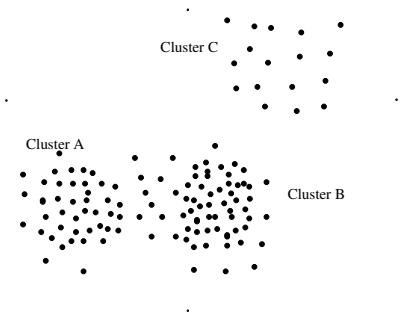


FIGURE 2.2 – Impact de la distribution locale sur les méthodes basées sur la densité

elles ne prennent pas le nombre de clusters comme paramètre. Cependant, pour utiliser ces approches il faut spécifier un seuil de densité $MinPts$ qui n'est pas intuitif et fixer un paramètre de voisinage avec un rayon Eps ou donner la résolution de la grille p^d. Ces paramètres influencent beaucoup la qualité du clustering surtout dans le cas où la densité des clusters est variées (pas homogène) comme le montre la figure 2.2. Dans cette example, si le seuil de densité est très élevé le cluster C ne peut pas être détecté. Par contre, si le seuil est très bas les deux clusters A et B seront fusionnés. Les algorithmes basés sur les distances comme le k-$means$ se comportent mieux pour cet example.

2.3.4 Algorithmes basés sur le clustering des noeuds d'un graphe

Les méthodes basées sur les graphes fournissent un cadre général dans lequel tous les types de données peuvent être partitionnés. L'idée de base est d'abord de transformer les données en un graphe de voisinage en utilisant une mesure de similarité (ou distance) appropriée au type de données. Ensuite, utiliser un algorithme de clustering de graphes (ou détection de communautés) pour extraire les partitions.

Étant donné un ensemble d'objets $\mathcal{O}$ et une mesure de similarité s, le graphe de voisinage $\mathcal{G} = (N, A)$ est construit comme suit :

— un nœud $n \in N$ est défini pour chaque objet $o \in \mathcal{O}$.
— une arête est ajoutée entre deux nœuds n, n' si et seulement si la distance entre les objets correspondants, $d(o, o')$, est inférieure à un seuil de voisinage v. Le poids d'une arête entre les deux nœuds n, n' est égal à la distance entre o et o', c-à-d, $w_{nn'} = d(o, o')$.

Après création du graphe de voisinage $\mathcal{G}$, un algorithme spécialisé dans le clustering des nœuds d'un graphe ou dans la détection de communauté peut être utilisé [XYFS07]. L'algorithme 11 résume le cadre

général de cette approche.
Notons que cette méthode basée sur la transformation des données en un graphe d'adjacence arrive à détecter des clusters de formes arbitraires parce qu'elle se base sur la densité pour définir les arêtes.

Algorithm 11: Algorithme basé les graphes

Input: Un ensemble de points $\mathcal{O} = \{o_1, \ldots, o_n\}$, un seuil de voisinage v, mesure de distance d
Output: Une partition P

```
1 G=CréerGraphe(O, v, d);
2 P= ClusteringGraphe(G) ;  /* ou un algorithme de détection de communauté */
3 return P;
```

2.4 Clustering sous contraintes

Un des problèmes importants du clustering est la variété des solutions obtenues par l'exécution des différents algorithmes sur un même ensemble de données. De plus, les qualités de ces solutions sont ordonnées différemment par rapport au différents critères de validation. La semi-supervision est proposée pour guider le processus de clustering en rajoutant des spécifications (contraintes). Le but du clustering sous contraintes et de trouver une partition des données qui satisfait toutes les contraintes spécifiées par l'utilisateur. Ces contraintes peuvent être classifiées en deux grandes catégories, des contraintes au niveau d'un cluster qui représentent les exigences sur les clusters (par exemple la taille) et des contraintes au niveau des objets qui spécifient les exigences sur des paires d'objets.

Plusieurs contraintes sur les objets ont été proposées et nous citons comme exemples :

— *Contraintes sur des objets singletons* : Ces contraintes donnent des informations sur la catégorie d'un objet (son cluster).
— *Contraintes « Must-link »* [WC00] : Ces contraintes sont appliquées sur des paires d'objets pour spécifier qu'ils sont de la même catégorie. Formellement, une contrainte « Must-link » entre deux objets o_i, o_j, notée $ML(o_i, o_j)$, signifie que o_i appartient à un cluster $\mathcal{C}$ si et seulement si o_j est dans $\mathcal{C}$.
— *Contraintes « Cannot-link »* [WC00] : Contrairement à la contrainte « Must-link », ces contraintes sont appliquées sur des paires d'objets pour spécifier qu'ils ne sont pas de la même catégorie. Formellement, une contrainte « Cannot-link » entre deux objets o_i, o_j, notée $CL(o_i, o_j)$, signifie que o_i peut appartenir à un cluster $\mathcal{C}$ si et seulement si o_j n'est pas dans $\mathcal{C}$.
— *Contrainte d'écart minimum entre clusters* [DR05] : Cette contrainte, notés θ-contrainte, exprime que la distance entre toute paire d'objets (o_i, o_j) qui appartiennent à des clusters différents, doit être au minimum égale à θ. Intuitivement, cette contrainte fixe l'écart minimum entre clusters ($splitP(P)$) voir la section 2.2.2.
— *λ-Contrainte* [DR05] : Cette contrainte spécifie que pour tout objet o_i dans un cluster $\mathcal{C}$, il existe au moins un autre objet $o_j \in \mathcal{C}$ tel que la distance $d(o_i, o_j) \leqslant \lambda$. Cette contrainte capture la notion de densité utilisée par *DBSCAN* (voir section 2.3.3).

Notons que la contrainte d'écart minimum entre clusters peut être réécrite avec une conjonction de contraintes « Must-link » entre toutes les paires d'objets qui ont une distance inférieure à θ. De la même façon, la λ-Contrainte est équivalente à une disjonction de contraintes « Must-link » entre toutes les paires d'objets qui ont une distance inférieure à λ. Il est à noter que trouver une partition avec des

contraintes « Cannot-link » ou une conjonction de contraintes « Must-link » et λ-contrainte est un problème $NP - complet$ [DR07].

En plus des contraintes exprimées sur les objets, des contraintes sur les clusters peuvent être spécifiées comme :

— *Capacité maximum* α : Cette contrainte spécifie le nombre maximum α d'objets que peut avoir un cluster, c-à-d, $\forall \mathcal{C} \in P, |\mathcal{C}| \leqslant \alpha$.

— *Capacité minimum* β [DBB08] : Cette contrainte spécifie le nombre minimum β d'objets que peut avoir un cluster, c-à-d, $\forall \mathcal{C} \in P, |\mathcal{C}| \geqslant \beta$.

— *Équilibre minimum* γ [BG06] : Cette contrainte est utilisée pour avoir des clusters de tailles équilibrées. L'équilibre d'une partition $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ est assuré par la contrainte suivante :

$$\frac{\underset{1 \leqslant i \leqslant k}{min} |\mathcal{C}_i|}{\underset{1 \leqslant j \leqslant k}{max} |\mathcal{C}_j|} \geqslant \gamma$$

.

2.4.1 Algorithmes pour le clustering sous contraintes

Plusieurs algorithmes ont été proposés pour mettre en œuvre la notion de contraintes dans le clustering. Une adaptation du $k\text{-}means$ a été proposée dans [WCRS01] pour considérer les contraintes « Must-link » (ML) et « Cannot-link » (CL).

Algorithm 12: Algorithme $COP\text{-}k\text{-}means$

Input: Un ensemble d'objet $\mathcal{O} = \{o_1, \ldots, o_n\}$ et un nombre de clusters k, deux ensembles de contraintes ML et CL

Output: Une partition $P = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$

1 $R = (\overline{o}_1, \ldots, \overline{o}_k) =$ sélectionnerAleatoirement($\mathcal{O}$, k);
2 **while** R *change* **do**
  /* affecter les objets au plus proche centroide */
3 **for** $i = 1$ *à* n **do**
4 $j = \underset{j \in \{1,\ldots,k\}}{\arg\min}\ d(o_i, \overline{o}_j)$;
5 **while** $\exists o \in \mathcal{C}_j$ *tel que* $(o_i, o) \in CL$ *et* $\exists o \notin \mathcal{C}_j$ *tel que* $(o_i, o) \in ML$ *et* $j \neq -1$ **do**
6 $j = \underset{j \in \{1,\ldots,k\}}{\arg\min}\ d(o_i, \overline{o}_j)$; /* l'ancien choix est supprimé */
7 **if** $j \neq -1$ **then**
8 $\mathcal{C}_j = \mathcal{C}_j \cup \{o_i\}$;
9 **else**
10 **return** échec ;
  /* mettre à jour les centroides */
11 **for** $i = 1$ *à* k **do**
12 miseAjour($\overline{o}_i$, $\mathcal{C}_i$);
13 **return** P;

La différence entre l'algorithme $k\text{-}means$ et $COP\text{-}k\text{-}means$ est dans la phase d'affectation des objets vers les clusters les plus proches. $COP\text{-}k\text{-}means$ affecte un objet o au plus proche cluster qui satisfait les contraintes ML et CL (ligne 8). L'algorithme commence par chercher le plus proche cluster

(ligne 4). Ensuite, si les contraintes ne sont pas violées l'objet est affecté (ligne 7) et l'algorithme continue l'affectation, sinon un autre choix de cluster est effectué jusqu'à satisfaction des contraintes. Si tous les clusters violent les contraintes ($j = -1$) l'algorithme s'arrête avec un échec (ligne 10). L'algorithme $COP\text{-}k\text{-}means$ essaye de satisfaire les contraintes à chaque itération sans faire des retours arrières. En effet, il se peut qu'il échoue à trouver une partition même si elle existe.

Pour échapper aux inconvénients de cet algorithme, une autre version de $k\text{-}means$ a été proposée dans [DR05]. L'étape clé de cette adaptation consiste à créer une nouvelle fonction objectif qui pénalise la violation des contraintes « Must-link » et « Cannot-link ». De la même façon, une nouvelle formule pour calculer les centroïdes a été proposée telle que si une contrainte $ML(o, o')$ est violée, le centroïde du cluster contenant o se rapproche du centroïde du cluster contenant o'. Par contres, la violation d'une contrainte $CL(o, o')$ oblige le centroïde contenant les deux objets à se déplacer vers le centroïde du cluster le plus proche. Une même approche a été proposée dans [PB07] en changeant la fonction objectif.

Une autre idée proposée dans [XJRN03] consiste à modifier la mesure de distance de telle sorte que deux objets liés par une contrainte « Must-link » soient très proches et les objets liés par une contrainte « Cannot-link » soient éloignés. Ce changement de la distance augmente la chance que les objets liés par des contraintes « Must-link » soient dans un même cluster et les objets liés par des contraintes « Cannot-link » soient dans des clusters différents.

Pour plus de détails sur les algorithmes de clustering sous contraintes nous référons les lecteurs vers le livre [BDW08].

2.4.2 Approches basées sur la programmation par contraintes

Dans [DRS10b] une méthode basée sur le problème 2-SAT a été proposée pour le clustering par contraintes avec 2 clusters seulement. Cette méthode optimise un critère de diamètre ou un critère de division. L'idée principale est d'encoder les contraintes avec des clauses binaires et utiliser des appels itératifs à un solveur 2-SAT pour optimiser le critère donné. Cette approche à deux grands avantages. Premièrement, elle permet d'avoir des algorithmes polynomiaux pour un clustering à deux clusters (k=2) car le problème 2-SAT est une classe polynomiale du problème SAT (voir chapitre 1). Deuxièmement, Elle garantit la satisfaction de toutes les contraintes et de trouver la meilleure solution si elle existe.

Pour encoder les contraintes en CNF, chaque objet $o_i \in \mathcal{O}$ est représenté par une variable propositionnelle x_i tel que si $x_i = 0$ alors l'objet correspondant o_i est affecté au premier cluster $\mathcal{C}_1$ sinon il est affecté au deuxième cluster $\mathcal{C}_2$.

La contrainte « Must-link » entre deux objets $ML(o, o')$ est encodée par les deux clauses suivantes :

$$(\neg x_i \vee x_j) \wedge (x_i \vee \neg x_j) \tag{2.28}$$

La première clause exprime que si $x_i = 1$ ($o_i \in \mathcal{C}_2$), alors la deuxième variable doit être affecté à vrai $x_j = 1$ ($o_j \in \mathcal{C}_2$). Par conséquent les deux objets sont dans le deuxième cluster. Par contre, si $x_i = 0$ ($o_i \in \mathcal{C}_1$), alors la deuxième variable doit être affecté à faux pour satisfaire la deuxième clause. Par conséquent les deux objets sont dans le premier cluster.

De même, la contrainte « Cannot-link » entre deux objets $CL(o_i, o_j)$ est encodée par les deux clauses suivantes :

$$(\neg x_i \vee \neg x_j) \wedge (x_i \vee x_j) \tag{2.29}$$

La contrainte de diamètre maximum d'un cluster est encodée en utilisant une contrainte « Cannot-link » entre toute paire d'objets (o_i, o_j) qui ont une distance supérieure à un seuil α, c-à-d, $d(o_i, o_j) > \alpha$. De la même façon, la contrainte d'écart minimum (splitP) est encodée en utilisant des contraintes « Must-link » entre tous les objets o_i et o_j qui ont une distance $d(o_i, o_j) < \theta$ avec θ est un seuil donné.

La formule CNF obtenue après l'encodage de toutes les contraintes est 2-SAT. La solution donnée par un solveur SAT est une partition des objets en deux clusters . Pour optimiser le critère de diamètre, l'algorithme proposée dans [DRS10b] utilise plusieurs appels au solveur SAT de tel sorte qu'à chaque itération la solution s'améliore en diminuant le seuil α pour le critère de diamètre et en augmentant le seuil θ pour le critère d'écart minimum. La solution optimale est obtenue à l'avant dernière itération avant que le solveur SAT retourne insatisfiable.

Dans [DDV13], les auteurs proposent une approche déclarative et générique basée sur la programmation par contraintes pour le clustering sous contraintes pour n'importe quel valeur de k. Elle permet de spécifier les différents critères d'optimisation ainsi que les différentes contraintes sur les clusters ou sur les paires d'objets. Le problème est modélisé sous forme d'un problème de satisfaction de contraintes (CSP) avec une fonction objectif à minimiser (ou maximiser).

Trois types de variables sont introduites. Une variable entière I_i est associée à chaque cluster $\mathcal{C}_i$ pour désigner son représentant. Le domaine de I_i est $dom(I_i) = \{1, \ldots, n\}$ où n est le nombre d'objets. Une variable G_i est associée à chaque objet (point) $o_i \in \mathcal{O}$ pour désigner son cluster. Le domaine de G_i est $dom(G_i) = \{1, \ldots, n\}$. Un objet o_i est dans un cluster $\mathcal{C}_j$ si et seulement si $G_i = I_j$. Enfin, la variable D est introduite pour représenter le diamètre maximal. C'est une variable entière avec un domaine formé par la distance minimale et la distance maximale entre chaque paire d'objets (points).

Une solution de clustering est une affectation complète des variables I, G et D qui satisfait les contraintes suivantes :

— chaque représentant appartient à son cluster $\forall i \in \{1, \ldots, k\}, G_{I_i} = I_i$;
— chaque objet est affecté à un cluster $\forall i \in \{1, \ldots, n\}, \bigvee_{j \in \{1,\ldots,k\}} G_i = I_j$;
— le représentant doit être d'indice minimal $\forall i \in \{1, \ldots, n\}, G_i \leqslant i$.

Les contraintes suivantes permettent d'éviter la symétrie entre les solutions (rotation des clusters) :

— le premier objet est le représentant du premier cluster $I_1 = 1$;
— les représentants sont en ordre croissant $\forall i < j \in \{1, \ldots, k\}, I_i \leq I_j$.

En utilisant ce CSP, plusieurs fonctions objectifs et contraintes ont été modélisées et nous pouvons citer comme exemple la minimisation du diamètre maximale :

— Diamètre : pour optimiser le diamètre les contraintes $\forall i < j \in \{1, \ldots, n\}, d(o_i, o_j) > D \rightarrow G_i \neq G_j$ sont ajoutées avec la fonction *minimiser* D.

De la même manière d'autres critères comme la somme des erreurs quadratiques (ssP) et les contraintes sur la taille des clusters peuvent être exprimées et ajoutées facilement. Pour améliorer la résolution du problème de clustering sous contraintes qui optimise le critère de la somme des erreurs quadratiques en utilisant l'approche CP, une contrainte globale qui gère ce critère a été proposée dans [DDV15]. Un algorithme de filtrage dédié a été proposé pour cette contrainte.

D'autres contributions récentes sur le clustering sous contraintes en utilisant CP on été proposées dans [TTBHCKC16, KRD+17, DDV17].

2.5 Conclusion

L'objectif de ce chapitre est de présenter le clustering et montrer le lien fort qui existe entre les mesures de distances et les algorithmes de clustering vis à vis du type de données. Pour cette raison, nous avons commencé par la présentation de différentes mesures de distance et de similarité qui ont été proposées pour différents types de données. Ensuite, nous avons décrit plusieurs mesures de qualité utilisées dans différentes approches de clustering. Enfin, nous avons présenté quelques approches de clustering qui se basent sur différents principes, à savoir, les approches hiérarchiques, les approches par partitionnement, les approches basées sur la densité et les approches basées sur les graphes. Un aperçu globale du problème de clustering sous contraintes a été fourni à la fin de ce chapitre. Les différentes notions présentées dans ce chapitre sont nécessaires pour comprendre les contributions proposées dans le quatrième chapitre.

Chapitre 3

Fouille de règles d'association

Sommaire

L'extraction des règles d'association est l'une des tâches fondamentales de la fouille de données. Elle vise à découvrir des relations intéressantes cachées dans de grandes bases de données. Ces relations entre les items (ensembles d'attributs) sont présentées sous forme d'implications, appelées règles d'association. Depuis la première application, largement connue sous la dénomination de panier de la ménagère [AS94], plusieurs nouveaux domaines d'application ont été identifiés comme la bio-informatique [AGF+09], le diagnostic médical [KW14], la détection d'intrusion [MD12], la fouille du web [Kaz09a] et l'analyse des données scientifiques [LS16]. Ce large spectre d'applications a permis à la fouille de règles d'association d'être appliquée à une variété de bases de données, y compris les données séquentielles, spatiales, et les données basées sur les graphes.

Dans ce chapitre, nous allons commencer par la définition du problème de fouille de règles d'association. Puis, nous décrivons les plus connues des algorithmes *ad-hoc* qui ont été proposés pour extraire ces règles. Ensuite, nous présentons d'autres variantes de règles qui ont été définies pour extraire un sous-ensemble de règles d'association jugées plus importantes, ou pour capturer d'autres régularités qu'on ne peut pas trouver à partir des règles standard. Enfin, nous présentons les approches déclaratives proposées récemment pour résoudre plusieurs problèmes de fouille de données transactionnelles.

3.1 Extraction de motifs ensemblistes

Étant donné un ensemble fini et non vide d'items (symboles), noté Ω, un motif ensembliste (appelé *itemset*) I sur Ω est défini comme étant un sous-ensemble non vide de Ω, c-à-d, $I \subseteq \Omega$. Un k-itemset est

Tid	Itemset
1	a b c d
2	a b e f
3	a b c
4	a c d f
5	g
6	d
7	d g

Tid	a	b	c	d	e	f	g
1	1	1	1	1	0	0	0
2	1	1	0	0	1	1	0
3	1	1	1	0	0	0	0
4	1	0	1	1	0	1	0
5	0	0	0	0	0	0	1
6	0	0	0	1	0	0	0
7	0	0	0	1	0	0	1

TABLE 3.1 – Représentation ensembliste et binaire d'une base de données transactionnelle $\mathcal{D}$.

un itemset qui contient exactement k items.

Le problème de fouille des itemsets est défini sur une base de données transactionnelles, notée $\mathcal{D}$, telle que $\mathcal{D}$ est définie comme étant un ensemble contenant n transactions, notées $t_1, \ldots, t_n$. Chaque transaction t_i est une paire (i, I) où i est un *identifiant* et I est un itemset. Notons qu'une base de données transactionnelles peut être vue comme une matrice binaire où chaque transaction est un vecteur binaire sur l'ensemble des items (voir Table 3.1).

Dans la suite de ce chapitre, nous nous intéressons aux motifs ensemblistes d'items. Ainsi, si aucune précision n'est apportée, un motif est un itemset. Nous utilisons les lettres majuscules I, J, K, etc pour désigner les itemsets et les lettres minuscules a, b, c, etc, pour désigner les éléments de Ω (items). De plus, les itemsets seront désignés par des chaînes pour alléger l'écriture (par exemple ab au lieu de $\{a, b\}$).

La pertinence d'un itemset est généralement évaluée via des critères fixés par l'utilisateur. Par exemple, le pourcentage de transactions qui contiennent un itemset I donne une estimation de sa fréquence. Cette dernière est obtenue en divisant le nombre de transactions couvrant I, appelé aussi Support, par la taille de la base.

Définition 3.1 (Couverture)**.** *Soit $\mathcal{D}$ une base de données transactionnelles. La couverture d'un itemset I dans $\mathcal{D}$, notée $C(I, \mathcal{D})$, est un sous-ensemble de transactions qui contiennent l'itemset I. Formellement, $C(I, \mathcal{D}) = \{(i, J) \in \mathcal{D} | I \subseteq J\}$.*

Définition 3.2 (Support)**.** *Soit $\mathcal{D}$ une base de données transactionnelles. Le support d'un itemset I dans $\mathcal{D}$, noté $S(I, \mathcal{D})$, est le nombre de transactions qui contiennent l'itemset I. Formellement, $S(I, \mathcal{D}) = |C(I, \mathcal{D})|$.*

Exemple 3.1. *Considérons comme exemple la base de données transactionnelles $\mathcal{D}$ présentée dans la Table 3.1. Dans ce cas, $C(ab, \mathcal{D}) = \{1, 2, 3\}$ et par conséquent $S(ab, \mathcal{D}) = 3$.*

Définition 3.3 (Itemsets fréquents)**.** *Étant donné une base de données transactionnelles $\mathcal{D}$ et un seuil minimum de support α, le problème de fouille d'itemsets fréquents, noté $FI(\mathcal{D}, \alpha)$, consiste à énumérer tous les itemsets I qui ont un support supérieur ou égal à α. Formellement, $FI(\mathcal{D}, \alpha) = \{I | I \subseteq \Omega$ et $S(I, \mathcal{D}) \geqslant \alpha\}$.*

Exemple 3.2. *Soit $\mathcal{D}$ la base de données transactionnelles représentée par la table 3.1. Pour un support minimum égal à 3, $FI(\mathcal{D}, 3) = \{a, b, c, d, ab, ac\}$*

Notons que si un itemset I apparaît dans une transaction t_i, alors t_i contient tous les sous-ensemble de I. En effet, la couverture de I est incluse dans la couverture de tout itemset $J \subset I$.

Propriété 3.1. *Soient $\mathcal{D}$ une base de données transactionnelles et α un seuil minimum de support. Soient I et J deux itemsets. Si $J \subseteq I$ alors $S(\mathcal{D}, J) \geqslant S(\mathcal{D}, I)$. Par conséquent, si $I \in FI(\mathcal{D}, \alpha)$ alors $J \in FI(\mathcal{D}, \alpha)$.*

La propriété 3.1 exprime que les sous-ensembles d'un itemset fréquent sont tous fréquents. Par conséquent, la contrainte de support est anti-monotone. Cette propriété est très importante et très utilisée par les algorithmes d'extraction des itemsets fréquents car elle permet l'élagage de l'espace de recherche et augmente l'efficacité de ces algorithmes.

3.1.1 Algorithmes pour l'extraction de motifs ensemblistes

Le problème de fouille d'itemset fréquents est un problème fondamental en fouille de données. Il a été présenté pour la première fois par Agarwal et al dans [AS94] comme une étape importante pour la fouille de règles d'association. Depuis, plusieurs autres domaines d'applications ont été identifiés [Agg15]. En effet, extraire efficacement l'ensemble des itemsets fréquents est devenu un problème fondamental en fouille de données. Nous présentons dans cette section les plus connus des algorithmes qui ont été proposés pour la fouille des itemsets fréquents.

Algorithm 13: Algorithme *Apriori*

Input: Une base de données transactionnelles $\mathcal{D}$ et un seuil minimum de support α
Output: Ensemble des itemsets fréquents FI

```
1  k = 1;
2  FI_k = { 1-itemsets fréquents} ;  /* trouver les items qui sont fréquents */
3  while FI_k ≠ ∅ do
4  |  FI_{k+1} = ∅;
5  |  C_{k+1} =GénérerCandidats(FI_k);
6  |  for tout I ∈ C_{k+1} do
7  |  |  S(I) = CalculSupport(D, I);
8  |  |  if S(I) ⩾ α then
9  |  |  |  FI_{k+1} = FI_{k+1} ∪ {I} ;
10 |  k = k + 1;
11 FI = ∪_k FI_k
12 return FI
```

Apriori **:** L'algorithme *Apriori* est le premier algorithme proposé pour l'extraction des itemsets fréquents [AS94]. L'idée de base de cet algorithme consiste à générer à chaque niveau k tous les k-itemsets fréquents à partir des itemsets fréquents du niveau précédent $k-1$. L'algorithme 13 présente les étapes de base de l'*Apriori*. Il commence par l'identification des items fréquents (ligne 1). Ensuite, à chaque itération k l'ensemble $\mathcal{C}_{k+1}$ qui contient les itemsets candidats de taille $k+1$ est généré en utilisant l'ensemble FI_k des itemset fréquent de taille k générée précédemment (lignes 5). Dans cette étape de génération de candidats, la propriété d'anti-monotonie de la contrainte du support minimum est utilisée pour éviter de générer des candidats à partir d'un itemset infréquent car un sur-ensemble d'un itemset qui n'est pas fréquent est forcément infréquent. Ceci permet une réduction significative de l'espace de recherche par rapport à une méthode qui explore tout l'espace des itemsets. Ensuite, le support de chaque

k-itemset candidat est calculé et les itemsets infréquents sont supprimés (lignes 6-9). Ce processus itératif continue jusqu'à ce qu'aucun itemset candidat ne puisse être généré.

L'algorithme $Apriori$ scanne toute la base de données pour calculer le support d'un itemset candidat. Cette opération est très coûteuse. Il y a eu des études approfondies sur les améliorations ou les extensions d'$Apriori$. Nous pouvons citer comme exemples, l'utilisation des techniques de hachage [PCY95] pour calculer efficacement le support d'un itemset candidat et l'utilisation de techniques de partitionnement proposée dans [SON95].

$FP-growth$ **:** Dans de nombreux cas, l'algorithme $Apriori$ réduit considérablement la taille des ensembles candidats. Cependant, il peut souffrir de :

— la génération d'un grand nombre de candidats ;
— le balayage répétitif de la base de données.

Han et al [HPY00] ont proposé une méthode, appelée $FP-growth$, qui extrait l'ensemble des itemsets fréquents sans génération de candidats. Le premier balayage de la base de données dérive une liste d'items fréquents donnés dans un ordre décroissant par rapport à leurs supports et supprime les items infréquents. Selon la liste décroissante des items, la base de données est compressée en un arbre de motifs fréquents « FP-tree » qui préserve les informations sur les itemsets. Cet arbre est fouillé en commençant par considérer chaque item comme suffixe initial. Puis construire une base de motifs conditionnel pour chaque suffixe (C'est l'ensemble des chemins de l'item vers la racine appelés préfixes). Ensuite construire pour chaque suffixe un arbre conditionnel et effectuer une extraction récursive sur cet arbre. La croissance du motif est obtenue par la concaténation du suffixe avec les motifs fréquents générés à partir de l'arbre conditionnel.

Pour chercher les motifs fréquents longs, l'algorithme $FP-growth$ recherche d'abord les motifs les plus courts, puis en concaténant les suffixes il obtient les plus longs. Pour plus de détailles voir [Agg15]

$Eclat$ **:** Les deux Algorithmes $Apriori$ et $FP-growth$ extraient les motifs fréquents à partir d'un ensemble de transactions présentées dans un format horizontal (c-à-d, identifiant, itemset). La table 3.1 est un exemple de cette représentation horizontale. Alternativement, la tâche de fouille d'itemsets fréquents peut être effectuée sur une base de données transactionnelles avec une représentation verticale (item, identifiants) où chaque item a une liste d'identifiants de transactions qui le contiennent. La table 3.2 illustre la représentation verticale de l'exemple présenté dans la table 3.1.
Dans [Zak00], Zaki a proposé l'algorithme $Eclat$ « Equivalence CLAss Transformation » qui utilise une représentation verticale des données. Le premier balayage de la base de données génère l'ensemble des identifiants des transactions de chaque item, c-à-d, construire la représentation verticale des données. Cette méthode est une recherche en largeur. En commençant par les itemsets de taille $(k = 1)$, les $(k+1)$-itemsets fréquents sont générés à partir des k-itemsets précédents comme dans l'algorithme $Apriori$ suivant un ordre sur les items. Le calcul de l'ensemble des identifiants de tout $(k+1)$-itemset est effectué par l'intersection des ensembles des identifiants des k-itemsets fréquents qui le génèrent. Ce processus est répété jusqu'à ce qu'aucun itemset fréquent ou aucun candidat ne puisse être trouvé.
L'avantage majeur de cette méthode est qu'elle ne scanne pas la base de données pour trouver le support d'un k-itemset candidat $(k > 1)$. Ceci parce que les ensembles des identifiants de chaque $(k-1)$-itemset portent l'information complète requise pour calculer le support du k-itemset.

items	identifiants
a	1 2 3 4
b	1 2 3
c	1 3 4
d	1 4 6 7
e	2
f	2 4
g	5 7

TABLE 3.2 – Une représentation verticale d'une base de données transactionnelles $\mathcal{D}$.

3.1.2 Représentations condensées

Les représentations condensées de motifs ont été proposées pour limiter le nombre de motifs extraits qui peut être très grand surtout pour des valeurs très petites du support minimum. L'idée de base est d'extraire un sous-ensemble de motifs à partir duquel tous les motifs peuvent être régénérés sans retour à la base de données. Ces représentations fournissent en partie une réponse au problème de la redondance. Un motif est redondant s'il peut être dérivé à partir d'autres motifs. Une représentation condensée permet en partie d'éviter les motifs redondants. De plus, elle facilite l'extraction des motifs et leur interprétation. On peut distinguer deux types de représentations condensées :

— *Exactes* : dans ce type de représentations la valeur exacte du support peut être régénérée pour chaque motif.
— *Approximatives* : ces représentations donnent une estimation de la valeur du support d'un motif.

De nombreuses représentations condensées ont été proposées pour les itemsets et on peut citer comme exemples : les itemsets fermés [PBTL99], les maximaux [BJ98] et les minimaux (appelés aussi générateurs) [BPT+00].

La qualité d'une représentation condensée est évaluée par rapport aux trois critères suivants :

1. *La taille de la représentation* : la taille de la représentation obtenue est un critère très important pour mesurer sa qualité. En effet, plus le nombre de motifs obtenus est petit plus la représentation est intéressante.
2. *L'efficacité de l'extraction* : l'extraction d'une représentation condensée doit être plus rapide que l'extraction de tous les motifs. Elle doit être efficace en temps et en espace mémoire utilisé. Par conséquent, plus l'extraction de la représentation condensée est rapide plus elle est intéressante.
3. *La complétude et l'efficacité de la régénération* : ce critère permet de passer d'une représentation condensée à l'ensemble complet de motifs sans aucune perte.

Itemsets fermés :

Définition 3.4. *Soit $\mathcal{D}$ une base de données transactionnelles. Un itemset I est fermé dans $\mathcal{D}$ si aucun de ses sur-ensembles n'a exactement le même support que lui, c-à-d, $\forall J \supset I$, $S(J,\mathcal{D}) < S(I,\mathcal{D})$.*

L'idée derrière cette représentation condensée vient du fait que les itemsets peuvent être regroupés suivant leurs couvertures dans des classes d'équivalences disjointes [PBTL99]. Chaque classe contient les itemsets qui ont la même couverture et donc forcément le même support. Ce regroupement en classes

disjointes suivant la couverture permet de préserver toutes les informations sans aucune perte. On peut remarquer que la couverture de chaque classe représente un rectangle de 1 maximal dans une représentation matricielle de la base de données transactionnelles. Maintenant pour compresser l'ensemble des itemsets, il suffit de représenter chaque classe par l'itemset maximal. Ces représentants de classes sont les itemsets fermés.

Définition 3.5 (Itemset fermé et fréquent)**.** *Soient $\mathcal{D}$ une base de données transactionnelles et α un seuil minimum de support. Un itemset I est fermé et fréquent dans $\mathcal{D}$ si et seulement si I est fermé et $S(I, \mathcal{D}) \geqslant \alpha$.*

En introduisant la contrainte de minimum support α, le problème de fouille d'itemsets fermés et fréquents dans une base de données transactionnelles $\mathcal{D}$, noté $CFI(\mathcal{D}, \alpha)$, consiste à énumérer tous les itemsets fermés et fréquents.

Exemple 3.3. *Prenons l'exemple de la table 3.1, pour un support minimum égal à 2 nous pouvons distinguer les classes d'équivalences suivantes : $\{a\}$, $\{b, ab\}$, $\{c, ac\}$, $\{d\}$, $\{f, af\}$, $\{g\}$,$\{bc, abc\}$, $\{ad, cd, acd\}$. Les itemsets fermés et fréquents sont donnés par l'ensemble $CFI(\mathcal{D}, 2)$=$\{a, ab, ac, d, af, g, abc, acd\}$.*

Notons que pour un seuil de support donné α, il suffit d'avoir tous les itemsets fermés et fréquents (l'ensemble $CFI(\mathcal{D}, \alpha)$) avec leurs couvertures, pour pouvoir générer tous les itemsets fréquents et leurs couvertures (l'ensemble $FI(\mathcal{D}, \alpha)$) sans retour vers la base de données. Soit $CFI(\mathcal{D}, \alpha)$ l'ensemble des itemsets fermés et fréquents dans $\mathcal{D}$ par rapport à un seuil de support α. L'ensemble exacte des itemsets fréquents $FI(\mathcal{D}, \alpha)$ est généré à partir de $CFI(\mathcal{D}, \alpha)$ en utilisant la règle suivante :

Soit I un itemset dont nous voulons retrouver sa couverture et son support :
- S'il n'existe pas un itemset fermé $J \in CFI(\mathcal{D}, \alpha)$ tel que $I \subseteq J$ alors I n'est pas fréquent, c-à-d, $I \notin FI(\mathcal{D}, \alpha)$.
- Sinon la couverture de I est celle du plus petit itemset fermé $J \in CFI(\mathcal{D}, \alpha)$ tel que $I \subseteq J$. En effet, $S(I, \mathcal{D}) = S(J, \mathcal{D})$.

L'exemple suivant illustre la régénération de la couverture d'un itemset à partir d'un ensemble d'itemsets fermés et fréquents.

Exemple 3.4. *Reprenons l'exemple 3.3. La couverture de l'itemset b est celle du plus petit itemset fermé contenant b ainsi, $C(b, \mathcal{D}) = C(ab, \mathcal{D})$. Par contre, il n'existe pas d'itemset fermé qui contient l'itemset bf et donc bf n'est pas fréquent par rapport à un seuil minimum de support égal à 2.*

Les itemsets fermés sont très importants. Il ont été utilisés dans différentes tâches de fouille de données comme l'extraction des règles d'association non redondantes [BPT+00] ou le clustering [DC03]. De nombreux algorithmes ont été proposés pour extraire l'ensemble des itemset fermés et fréquents efficacement. Nous pouvons citer comme exemples `A-close` [PBTL99] basé sur le principe d'*Apriori*, *Charm* [ZH02] qui utilise une représentation verticale de la base de données d'une manière similaire à *Eclat*, et $FP - Close$ [GZ03] qui utilise le même principe que $FP - growth$.

Itemsets fréquents et maximaux :

Définition 3.6. *Soient $\mathcal{D}$ une base de données transactionnelles et α un seuil minimum de support. Un itemset I est un itemset fréquent et maximal si et seulement si $S(I, \mathcal{D}) \geqslant \alpha$ et $\forall J$ tel que $I \subset J$, $S(J, \mathcal{D}) < \alpha$.*

Un itemset I est fréquent et maximal s'il n'existe pas un autre itemset fréquent J contenant I. Les itemsets maximaux constituent une bordure entre les itemsets fréquents et les itemsets infréquents. L'ensemble des maximaux dans une base de données $\mathcal{D}$, noté $MFI(\mathcal{D}, \alpha)$, est considéré comme une représentation condensée de l'ensemble des itemsets fréquents $FI(\mathcal{D}, \alpha)$. En effet, tous les éléments de $FI(\mathcal{D}, \alpha)$ peuvent être régénérés à partir de $MFI(\mathcal{D}, \alpha)$ en énumérant tous les sous-ensembles des itemsets maximaux. Cependant, cette représentation ne préserve pas l'information sur les couvertures des itemsets. Par conséquent, nous ne pouvons pas retrouver la couverture exacte d'un itemset qui n'est pas maximal. Cette représentation est très intéressante car sa taille est très réduite par rapport aux ensembles $FI(\mathcal{D}, \alpha)$ et $CFI(\mathcal{D}, \alpha)$.

Exemple 3.5. *Soient $\mathcal{D}$ la base de données transactionnelles présentée dans la table 3.1 et $\alpha = 2$ un seuil minimum de support. L'itemset abc est fréquent et maximal alors que ab est fréquent mais pas maximal car il a un sur-ensemble abc qui est fréquent. L'ensemble des itemsets fréquents et maximaux pour un support minimum $\alpha = 2$ est $MFI(\mathcal{D}, 2) = \{af, abc, acd\}$. Ainsi, il y a seulement 3 itemsets fréquents et maximaux pour $\alpha = 2$ alors qu'il y a 9 itemsets fermés et 14 itemsets fréquents. Ceci montre la compacité de l'ensemble des maximaux.*

Bayardo [BJ98] a étudié pour la première fois l'énumération des itemsets maximaux. Une méthode de recherche basée sur le principe d'$Apriori$ a été proposée pour trouver ces motifs. Cette approche est une recherche en largeur qui utilise l'anti-monotonie de la contrainte de support minimum pour élaguer et réduire l'espace de recherche. Une autre méthode efficace, appelée $MAFIA$, proposée par Burdick et al [BCG01]. Elle utilise des bitmaps verticaux pour compresser la liste des identifiants des transactions pour améliorer l'efficacité du comptage. Plusieurs autres algorithmes et implémentations efficaces ont été proposés pour extraire les itemsets maximaux comme lcm [UKA04].

Itemsets minimaux : Nous avons vu précédemment que les itemsets peuvent être regroupés en classes d'équivalences par rapport à leurs couverture. Contrairement à un itemset fermé, un itemset minimal [BPT+00, BBR00] (appelé aussi itemset libre ou générateur) est le plus petit itemset dans une classe d'équivalence (au sens de l'inclusion ensembliste). Notons que contrairement aux itemsets fermés, on peut trouver plusieurs itemsets minimaux dans une même classe d'équivalence. Par conséquent l'ensemble des itemset minimaux est généralement plus grand que l'ensemble des fermés.

Définition 3.7 (itemset minimal)**.** *Soient $\mathcal{D}$ une base de données transactionnelles et I un itemset. I est minimal dans $\mathcal{D}$ si et seulement s'il n'existe pas d'itemset $J \subset I$ et $S(I, \mathcal{D}) = S(J, \mathcal{D})$.*

Définition 3.8 (itemset minimal et fréquent)**.** *Soient $\mathcal{D}$ une base de données transactionnelles et α un seuil minimum de support. Un itemset I est minimal et fréquent dans $\mathcal{D}$ si et seulement si $S(I, \mathcal{D}) \geqslant \alpha$ et I est un itemset minimal.*

L'ensemble des itemsets fréquents minimaux dans une base de données $\mathcal{D}$ par rapport un seuil de support α, noté $GFI(\mathcal{D}, \alpha)$, est une représentation condensée exacte. Ce type de motifs a été utilisé pour définir des sous-ensembles de règles d'association [BPT+00, Zak04]. Il est important de noter qu'une généralisation de cette représentation, appelés motifs δ-libres, a été proposée par Boulicault et al dans [BBR00]. Pour $\delta = 0$, nous obtenons le cas particulier des itemsets minimaux.

Exemple 3.6. *Reprenons l'exemple de la table 3.1. L'ensemble des motifs minimaux pour un support minimum égal à 2 est $GFI(\mathcal{D}, 2) = \{a, b, c, d, f, g, bc, ad, cd\}$. L'itemset ab n'est pas minimal car l'itemset b a le même support que ab. Par contre, l'itemset ad est minimal car le supports de a et le support de d sont plus grands que le support de ad.*

Plusieurs autres représentations condensées ont été proposées dans la littérature. Un aperçu sur la majorité de ces dernières a été présenté dans [CRB06].

3.2 Règles d'association

L'extraction des règles d'association est une tâche fondamentale en fouille de données [AS94]. Elle vise à découvrir des relations intéressantes cachées dans de grandes bases de données transactionnelles. Ces relations entre les items sont présentées sous forme d'implications. Considérons par exemple une base de données sur les paniers de consommation où chaque transaction représente les articles (items) achetés par un client. Dans ce cas, il est intéressant pour le gérant du magasin de connaître le fait que 80% des clients ayant acheté des céréales et du sucre ont également acheté du lait. Le problème de fouille de règles d'association a été défini pour extraire ce genre d'informations cachés.

3.2.1 Extraction des règles d'association

Les itemsets fréquents sont utilisés pour générer les règles d'association de la forme $X \rightarrow Y$ où X, appelé antécédent, et Y, appelé conséquence, sont deux itemsets. Nous utilisons dans la suite de cette section les lettres X (respectivement Y) pour noter l'antécédent (respectivement la conséquence) d'une règle. Le support et la confiance sont deux mesures statistiques très importantes pour mesurer la pertinence d'une règle donnée. En effet, pour une règle d'association $r : X \rightarrow Y$ le support permet de voir le nombre de transactions qui contiennent les deux itemsets ensembles $(X \cup Y)$. Par contre, la confiance est la probabilité conditionnelle qu'une transaction contenant l'itemset X contienne aussi l'itemset Y.

Définition 3.9 (Couverture d'une règle d'association)**.** *Soit $\mathcal{D}$ une base de données transactionnelles. La couverture d'une règle $X \rightarrow Y$ dans $\mathcal{D}$ est égale à la couverture de l'itemset $X \cup Y$.*

$$C(X \rightarrow Y, \mathcal{D}) = C(X \cup Y, \mathcal{D})$$

Définition 3.10 (Support d'une règle d'association)**.** *Soit $\mathcal{D}$ une base de données transactionnelles. Le support d'une règle $X \rightarrow Y$ dans $\mathcal{D}$ est défini par la mesure suivante :*

$$S(X \rightarrow Y, \mathcal{D}) = |C(X \rightarrow Y, \mathcal{D})|$$

Définition 3.11 (Confiance d'une règle d'association)**.** *Soit $\mathcal{D}$ une base de données transactionnelles. La confiance d'une règle $X \rightarrow Y$ dans $\mathcal{D}$ est définie comme suit :*

$$Conf(X \rightarrow Y, \mathcal{D}) = \frac{S(X \rightarrow Y, \mathcal{D})}{S(X, \mathcal{D})}$$

Exemple 3.7. *Considérons la base de données transactionnelles représentée dans la table 3.1. La couverture de la règle d'association $ab \rightarrow c$ est l'ensemble $C(ab \rightarrow c, \mathcal{D}) = \{1, 3\}$. Son support est $S(ab \rightarrow c, \mathcal{D}) = 2$ et sa confiance $Conf(ab \rightarrow c, \mathcal{D}) = \frac{2}{3}$*

Comme pour le cas du support, un seuil de confiance minimum β est utilisé pour générer les règles d'association les plus pertinentes. Une règle d'association est définie en utilisant les deux contraintes de support minimum et de confiance minimum comme suit :

Définition 3.12 (Règle d'association)**.** *Soient X, Y deux itemsets. La règle $X \rightarrow Y$ est dite règle d'association sur une base de données transactionnelles $\mathcal{D}$ pour un seuil minimum de support α et un seuil minimum de confiance β si et seulement si elle satisfait les deux critères suivants :*

1. *le support de l'itemset $X \cup Y$ est supérieur à α, c-à-d, $S(X \rightarrow Y, \mathcal{D}) \geqslant \alpha$;*
2. *la confiance de $X \rightarrow Y$ est au minimum β, c-à-d, $Conf(X \rightarrow Y, \mathcal{D}) \geqslant \beta$.*

Le premier critère de la définition 3.12 assure qu'un nombre suffisant de transactions vérifie la règle alors que le deuxième critère assure que la règle est assez forte en terme de probabilité conditionnelle .

Exemple 3.8. *Soit $\mathcal{D}$ la base de données représentée par la table 3.1. Pour un seuil minimum de support $\alpha = 3$ et un seuil minimum de confiance $\beta = 1/2$, la règle $a \rightarrow b$ est une règle d'association tandis que $b \rightarrow c$ ne l'est pas car $S(b \rightarrow c, \mathcal{D}) = 2 < \alpha$.*

La méthode standard pour générer les règles d'association utilise deux étapes principales. Ces étapes correspondent aux deux critères de la définition 3.12 qui représentent les contraintes de support et de confiance.

1. *Étape 1* : extraction de tous les itemsets fréquents pour un support minimum α.
2. *Étape 2* : génération de toutes les règles d'association à partir de l'ensemble des itemsets fréquents par rapport à un seuil minimum de confiance β.

La première étape est plus difficile que la deuxième. En effet, la majorité des algorithmes proposés pour l'extraction des règles d'association se focalisent sur l'extraction efficace des itemsets fréquents. Nous avons présenté précédemment dans la section 3.1.1 le principe de fonctionnement de quelques algorithmes. Supposons maintenant que l'ensemble des itemsets fréquents FI est trouvé. Alors une façon simple d'extraire les règles confiantes est de partitionner tout itemset $I \in FI(\mathcal{D}, \alpha)$ en toutes combinaisons possible X et $Y = I \setminus X$ tel que $I = X \cup Y$. Ensuite la confiance de chaque règle $X \rightarrow Y$ est mesurer pour décider si la règle est pertinente ou non.

Il est important de noter que les règles d'association satisfont la propriété d'anti-monotonie de la contrainte de confiance. Elle est similaire à celle du support et peut être utilisée pour accélérer la deuxième étape du processus d'extraction des règles d'association.

Propriété 3.2. *Soient $\mathcal{D}$ une base de données transactionnelles et X, Y, Z trois itemsets tel que $X \subset Y \subset Z$. Alors la confiance de la règle $Y \rightarrow Z \setminus Y$ est au moins égale à la confiance de la règle $X \rightarrow Z \setminus X$.*

$$Conf(Y \rightarrow Z \setminus Y), \mathcal{D}) \geqslant Conf(X \rightarrow Z \setminus X)$$

3.2.2 Variantes des règles d'associations

Les règles d'association sont des connaissances très importantes à découvrir à partir de grandes bases de données transactionnelles. Cependant, deux problèmes majeurs peuvent être rencontrés avec cet ensemble. Premièrement, le nombre de ces règles est souvent important, ce qui limite leur utilité dans des applications réelles. Deuxièmement, cet ensemble de règles ne permet pas de capturer des relations autres que les implications entre deux itemsets.

Pour résoudre le premier problème, deux types de solutions ont été proposés. Le premier consiste à utiliser des mesures statistiques pour décider de la pertinence d'une règle. Ceci permet l'extraction des règles les plus utiles. Le deuxième type consiste à extraire une représentation condensée de l'ensemble des règles d'association. Dans le cas idéal, la représentation doit être sans perte (permet la dérivation de l'ensemble des règles d'association), exacte (ne permet pas la dérivation des règles qui ne sont pas valides par rapport aux seuils de support et de confiance) et informative (permet la détermination des paramètres de chaque règle, c-à-d, le support et la confiance). Plusieurs représentations condensées de l'ensemble des règles d'association ont été proposées et nous citons par exemple, la base générique et la base informative [BPT+00], les règles représentatives [Kry98a], les règles fermées [Sza06], les règles minimales non redondantes [BPT+00] et l'ensemble des règles non redondantes présenté par Zaki dans

[Zak04]. Une présentation détaillée de plusieurs représentations condensées est donnée par Kryszkiewicz dans [Kry02]. Dans cette sous section nous présentons les trois représentations qui correspondent à nos contributions à savoir les règles fermées, les règles minimales non redondantes et les règles non redondantes de Zaki.

Pour résoudre le second problème et extraire des relations que nous ne pouvons pas avoir à partir de l'ensemble des règles d'association standard, d'autres types de règles ont été définis telles que les règles inattendues [PT99] et les règles d'exception [Suz02] qui cherchent à identifier des déviations par rapport à un ensemble de règles valides.

Règles fermées : Dans [PBTL99], les auteurs ont montré que l'ensemble des itemsets fermés est une représentation condensée sans perte de l'ensemble des itemsets fréquents. En se basant sur ce résultat, une représentation condensée de l'ensemble des règles d'association appelée règles d'association fermées, noté $\mathcal{CR}$, a été proposée dans [Sza06]. Cet sous-ensemble de règles est sans perte et il permet la régénération de l'ensemble de règles d'association avec leurs supports et confiances exactes sans retour à la base de données. De plus, le nombre de règles fermés est très réduit par rapport au nombre total de règles d'association surtout pour les bases de données denses.

Définition 3.13 (Règle d'association fermées)**.** *Soient $\mathcal{D}$ une base de données transactionnelles, α un seuil minimum de support et β un seuil minimum de confiance. Une règle d'association $X \rightarrow Y$ est une règle fermée si et seulement si l'itemset $X \cup Y$ est un itemset fermé et $X \rightarrow Y$ est une règle d'association valide par rapport à α et β, c-à-d :*

$$\mathcal{CR}(\mathcal{D}, \alpha, \beta) = \{X \rightarrow Y | X \cup Y \in CFI(\mathcal{D}, \alpha) \wedge S(X \rightarrow Y, \mathcal{D}) \geqslant \alpha \wedge Conf(X \rightarrow Y, \mathcal{D}) \geqslant \beta\}$$

Autrement dit, une règle $X \rightarrow Y$ est redondante s'il existe une règle $X' \rightarrow Y'$ différente de la première telle que $S(X \rightarrow Y, \mathcal{D}) = S(X' \rightarrow Y', \mathcal{D})$ et $X \subseteq X'$ et $Y \subseteq Y'$. En effet, la représentation par les règles fermées permet d'éliminer les règles redondantes qui sont incluses dans d'autres règles qui couvrent les mêmes ensembles de transactions.

L'ensemble des règles fermées a l'avantage d'être facilement calculable car l'extraction efficace des itemsets fermés est suffisante pour générer toutes les règles fermées. En effet, la méthode proposée dans [Sza06] pour l'extraction des règles fermées dans une base de données transactionnelles $\mathcal{D}$ suit les deux étapes suivantes :

1. générer l'ensemble des itemsets fermés et fréquents dans $\mathcal{D}$, c-à-d, $CFI(\mathcal{D}, \alpha)$;
2. à partir de chaque itemset fermé X, générer toutes les règles $X \setminus Y \rightarrow Y$ avec $Y \subset X$ qui vérifient la contrainte de confiance minimum β.

Pour calculer la confiance de la règle $X \setminus Y \rightarrow Y$, il faut avoir le support de $X \setminus Y$ qui peut ne pas être fermé. Comme l'ensemble des itemsets fermés est sans perte, alors le support de $X \setminus Y$ est égal à celui du plus petit itemset fermé qui le contient. En effet, l'ensemble des itemsets fermés est suffisante pour la génération de toutes les règles fermées.

Exemple 3.9. *Reprenons la base de données transactionnelles $\mathcal{D}$ de la table 3.1. Pour un seuil minimum de support $\alpha = 2$ et un seuil minimum de confiance $\beta = 1/2$, la règle $a \rightarrow d$ n'est pas une règle d'association fermée car $ac \rightarrow d$ est une règle d'association et $S(ac \rightarrow d, \mathcal{D}) = S(a \rightarrow d, \mathcal{D})$.*

Règles minimales non redondantes : Dans [BPT+00], les auteurs caractérisent un sous-ensemble de règles d'association, appelées règles d'association minimales non redondantes (MNR). Ils considèrent qu'une règle d'association est intéressante si elle contient le moins de conditions et le plus de conséquences. Autrement dit, une règle d'association est minimale non redondante s'il n'existe pas d'autre règle qui a moins de conditions et implique plus de conséquences qu'elle. L'idée derrière provient de la logique propositionnelle. En logique la formule $A \rightarrow B \wedge C$ permet la déduction des deux règles $A \rightarrow B$ et $A \rightarrow C$. En effet, l'implication $A \rightarrow B \wedge C$ est plus informative. Une règle minimale non redondante est définie comme suit.

Définition 3.14 (Règle minimale non redondante)**.** *Une règle d'association $X \rightarrow Y$ est minimale non redondante s'il n'existe pas de règle $X' \rightarrow Y'$ qui a le même support et la même confiance que $X \rightarrow Y$ et que $X' \subseteq X$ et $Y \subseteq Y'$.*

En d'autres termes, une règle $r : X \rightarrow Y$ est redondante si on peut rétrécir (resp . élargir) son antécédent (resp. sa conséquence) en supprimant (resp. ajoutant) des items de X (resp. à Y) sans changer la confiance (resp. le support) de r.

La représentation condensée de l'ensemble des règles d'association par l'ensemble des règles minimales non redondantes a été prouvée exacte (sans perte) [Kry02] car elle permet la régénération de toutes les règles avec leurs supports et confiances exactes.

Exemple 3.10. *Soit $\mathcal{D}$ la base de données transactionnelles de la table 3.1. La règle d'association $ad \rightarrow b$ n'est pas une règle minimale non redondante contrairement à la règle $ad \rightarrow bc$ car les deux règles ont le même support et la même confiance et $b \subset bc$.*

Pour extraire les règles minimales non redondantes, la méthode suivante à été proposée :

1. extraire l'ensemble des itemsets fermés $CFI(\mathcal{D}, \alpha)$;
2. extraire pour chaque itemset fermé $Y \in CFI(\mathcal{D}, \alpha)$ l'ensemble des itemsets minimaux (générateurs), noté $GI(Y, \mathcal{D})$, qui appartiennent à la même classe d'équivalence de Y, c-à-d, ils ont la même couverture ;
3. pour chaque itemset minimal X, identifier tous les itemsets fermés Y qui représentent des sur-ensembles de X et générer les règles minimales non redondantes de la forme $X \rightarrow Y \setminus X$ en respectant un seuil minimum de confiance β.

Notons que dans cette méthode, la minimalité d'une règle $X \rightarrow Y$ est capturée par le fait que X est un itemset minimal et la maximalité de Y est capturée par le fait que $X \cup Y$ est un itemset fermé.
Un algorithme efficace, appelé $Zart$, a été proposé dans [SNK06] pour extraire l'ensemble des règles minimales non redondantes en effectuant une extraction rapide de l'ensemble des itemsets fermés avec leurs générateurs minimaux.

Règles les plus générales : D'une façon similaire aux règles minimales non redondantes, Zaki [Zak04] propose d'extraire un sous-ensemble très réduit de règles d'association. Il considère qu'une règle est non redondante si elle a le moins de conditions (antécédents) et le moins de conséquences. Dans [Zak04], l'auteur n'a pas donné de caractérisation (ou définition) exacte de ce sous-ensemble de règles d'association. Il a proposé un algorithme basé sur l'extraction des itemsets fermés pour générer les règles non redondantes. Pour définir la redondance d'une règle, l'auteur a défini un sous-ensemble de règles d'association, appelées les règles les plus générales.

Définition 3.15 (Règle d'association plus générale)**.** *Soit $r_1 : X \rightarrow Y$, $r_2 : X' \rightarrow Y'$ deux règles d'association. On dit que la règle r_1 est plus générale que r_2 si et seulement si r_2 peut être générée à*

partir de r_1 en rajoutant des items, soit à l'antécédent X, soit à la conséquence Y, c-à-d, $X \subseteq X'$ et $Y \subseteq Y'$.

En utilisant la définition 3.15, Zaki a défini la redondance dans un ensemble de règles d'association qui ont la même couverture et la même confiance (classe d'équivalence) comme suit.

Définition 3.16 (Règle redondante)**.** *Soit $\mathcal{R}$ une classe d'équivalence de règles qui ont la même couverture et la même confiance. Une règle $r \in \mathcal{R}$ est dite redondante si et seulement s'il existe une règle $r' \in \mathcal{R}$ telle que r' est plus générale que r.*

En d'autres termes, pour chaque classe d'équivalence de règles, les règles les plus générales sont suffisantes pour représenter toutes les règles de cette classe car elles ont la même couverture et la même confiance. En effet, les règles non redondantes de Zaki sont définies par l'ensemble de règles les plus simples qui ont des antécédents et des conséquences minimaux.

Pour générer cet ensemble de règles non redondantes, la méthode proposée dans [Zak04] commence d'abord par extraire tous les itemsets fermés avec leurs générateurs minimaux. Ensuite un algorithme dédié à l'extraction des règles non redondantes est exécuté pour générer l'ensemble des règles non redondantes à partir de toute paire d'itemsets fermés X, Y tel que $X \subseteq Y$. L'algorithme 14 décrit l'extraction d'un ensembles de règles non redondantes à partir de deux itemsets fermés X et Y. Nous distinguons deux étapes principales, la première concerne l'extraction des règles avec une confiance de 100% qui vont d'un générateur de Y, noté X', vers un autre générateur égal à $X'' \setminus X'$ oú X'' est un générateur de X (ligne 4-7). De la même façon, la deuxième étape permet la génération des règles d'association avec des confiances inférieures à 100% qui vont d'un générateur minimal de X, noté X', vers un autre générateur minimal égale à $X'' \setminus X'$ où X'' est un générateur minimal de Y (ligne 8-11). Enfin, une étape de filtrage qui garde les règles les plus générale dans $\mathcal{R}$ est effectuée (ligne 12).

Il est à noter que Zaki [Zak04] a considéré que les itemsets fermés X et Y sont adjacents, c-à-d, qu'il n'existe pas un itemset fermé Z tel que $X \subset Z \subset Y$ pour éviter la transitivité dans l'ensemble des règles non redondantes. Cela réduit largement le nombre de règles extraites.

Exemple 3.11. *Soit $\mathcal{D}$ la base de données transactionnelles représentée dans la table 3.1. La règle $ab \rightarrow c$ est une règle redondante car la règle $b \rightarrow c$ est plus générale que $ab \rightarrow c$.*

Règles inattendues : Nous avons présenté précédemment quelques représentations condensées de l'ensemble de règles d'association. Maintenant, nous décrivons des variantes de règles qui permettent la détection d'un comportement inhabituel qui ne peut pas être capturé par des règles d'association standard et nous commençons par les règles inattendues. Ce type de règles a été introduit par Padmanabhan et al dans [PT99, PT98]. L'intuition derrière est de trouver toutes les règles d'association qui contredisent une croyance préalable donnée sous la forme d'une règle $U \rightarrow V$.

Définition 3.17 (Règle inattendue)**.** *Soient $\mathcal{D}$ une base de données transactionnelles, α un seuil minimum de support, β un seuil minimum de confiance et $U \rightarrow V$ une règle de croyance où U et V sont deux itemsets. $X \rightarrow Y$ est une règle inattendue si et seulement si les conditions suivantes sont satisfaites :*

- *l'itemset $Y \cup V$ n'est pas fréquent dans $\mathcal{D}$ par rapport à α, c-à-d, $S(Y \cup V, \mathcal{D}) < \alpha$;*
- *$X \cup U$ est un itemset fréquent dans $\mathcal{D}$, c-à-d, $S(X \cup U, \mathcal{D}) \geqslant \alpha$;*
- *$X \cup U \rightarrow Y$ est une règle d'association fréquente avec une confiance élevée, c-à-d, $S(X \cup U \rightarrow Y, \mathcal{D}) \geqslant \alpha$ et $Conf(X \cup U \rightarrow Y, \mathcal{D}) \geqslant \beta$;*

Algorithm 14: Algorithme de génération de règles non redondantes de Zaki

Input: X, Y deux itemsets fermés et fréquents dans $\mathcal{D}$ et $X \subseteq Y$
Output: Ensemble de règles non redondantes $\mathcal{R}$

```
1  G1 ← GénérateursMinimaux(X);
2  G2 ← GénérateursMinimaux(Y);
3  R = ∅;
   /* Règles avec une confiance de 100%                              */
4  for tout X' ∈ G2 do
5  |  for tout X'' ∈ G1 do
6  |  |  if C(X,D) = C(X' \ X'',D) et C(Y,D) = C(X' ∪ X'',D) then
7  |  |  |  R = R ∪ {X' → (X'' \ X')} ;
   /* Règles avec une confiance < 100%                               */
8  for tout X' ∈ G1 do
9  |  for tout X'' ∈ G2 do
10 |  |  if C(Y,D) = C(X' ∪ X'',D) then
11 |  |  |  R = R ∪ {X' → (X'' \ X')};
12 Filtrer(R);
13 return R;
```

— $X \cup U \rightarrow V$ *n'est pas une règle d'association, c-à-d,* $S(X \cup U \rightarrow V, \mathcal{D}) < \alpha$ *ou* $Conf(X \cup U \rightarrow V, \mathcal{D}) < \beta$.

Padmanabhan et al [PT98] ont appliqué l'extraction des règles inattendues sur une base de données transactionnelles réelle qui concerne l'historique des achats des clients. Ils ont découvert que pour la croyance suivante *"Les clients qui travaillent font plus d'achats les weekends qu'en semaine"*, La règle *"Au mois de décembre les clients font plus d'achats la semaine qu'en weekends"* est une règle inattendue. Un algorithme, appelé `ZoomUR` « Zoom Unexpected Rules », a été proposé pour extraire les règles inattendues par rapport à un ensemble de croyances données par l'utilisateur.

Règles d'exception : Les règles d'exception sont généralement présentées comme des motifs qui expriment une déviation par rapport à un comportement connu donné sous forme d'une règle d'association standard (exception). Ce type de règles a été proposé par Suzuki et al dans [Suz02]. Contrairement au règles inattendues, l'extraction des règles d'exception est automatique sans introduction de la règle de référence. La forme générale d'une règle d'exception est comme suit :

Définition 3.18. *Soient $\mathcal{D}$ une base de données transactionnelle, α un seuil minimum de support et β un seuil minimum de confiance et X, Y, Z trois itemsets avec $|Y| = 1$. Z est une exception si et seulement si les conditions suivantes sont satisfaites :*

1. $X \rightarrow Y$ *est une règle d'association valide dans $\mathcal{D}$ par rapport à α et β.*
2. $X \cup Z \rightarrow \neg Y$.
3. $X \rightarrow Z$ *n'est pas une règle d'association.*

Notons que l'itemset Y représente généralement un attribut de classe. En effet, la couverture de $\neg Y$ est égale au transactions qui ne contiennent pas Y (là où l'attribut n'apparaît pas).

L'interprétation d'une exception est comme suit : X implique fortement Y mais en présence de Z, X n'implique pas Y ou implique sa négation. Par exemple, si X représente les `antibiotiques`, Y

représente la `guérison` et Z représente le `staphylocoque` alors la règle d'exception suivante est extraite : « A l'aide des antibiotiques le malade va guérir, sauf si le `staphylocoque` apparaît alors le malade va mourir ».

Il est important de noter qu'un autre type de règles similaires aux règles d'exception, appelées règles d'anomalies, a été présenté dans [BCMG04].

3.3 Approches basées sur la programmation par contraintes pour l'extraction des motifs ensemblistes

Récemment, de nouvelles approches déclaratives ont été proposées pour traiter plusieurs tâches de fouille de données. Cette tendance de recherche a été initiée par Luc De Raedt et al dans [RGN08] où le le fameux problème de fouille d'itemsets fréquents est modélisé et résolu en utilisant la programmation par contraintes (PPC). Ce nouveau cadre offre un modèle de représentation déclaratif et flexible. En effet, de nouvelles contraintes nécessitent souvent de nouvelles implémentations pour les approches spécialisées, alors qu'ils peuvent être facilement intégrés dans un tel cadre basé sur PPC. Il permet aux problèmes de fouille de données de bénéficier de plusieurs techniques de résolution génériques et efficaces de PPC. Cette première étude conduit à la première approche basée sur PPC pour l'extraction des itemset en présentant des propriétés déclaratives très intéressantes. Ce travail a contribué à l'émergence d'une nouvelle tendance de recherche permettant une fertilisation croisée entre la fouille de données et l'intelligence artificielle. Encouragés par ces résultats prometteurs, plusieurs contributions ont abordé d'autres problèmes de fouille de données à l'aide des deux modèles très connus d'IA, à savoir la programmation par contraintes et la satisfiabilité propositionnelle comme la fouille des séquences [CJSS12, AGS16] et la fouille des graphes [JMRS17].

3.3.1 Modélisation du problème de fouille de motifs ensemblistes en CP et SAT

Nous présentons dans cette sous section la modélisation du problème de fouille d'itemsets fréquents en problème de programmation par contraintes proposé dans [RGN08].
Soient $\mathcal{D} = \{t_1, \ldots, t_m\}$ une base de données transactionnelle qui contient m transactions et Ω un ensemble d'items. Dans [RGN08], les auteurs ont remarqué que le problème de fouille d'itemsets fréquents pour un seuil minimum de support α consiste à énumérer l'ensemble suivant :

$$FI = \{(I, T) | I \subseteq \Omega, T \subseteq \mathcal{D}, T = C(I, \mathcal{D}), |T| \geqslant \alpha\}$$

Cette observation a été utilisée pour modéliser le problème d'extraction d'itemsets fréquents en un problème de programmation par contraintes en modélisant les deux contraintes $T = C(I, \mathcal{D})$ et $|T| \geqslant \alpha$ séparément. $T = C(I, \mathcal{D})$ représente la contrainte de couverture de I dans $\mathcal{D}$, tandis que $|T| \geqslant \alpha$ est la contrainte de support minimum.

Pour modéliser cette formalisation en CP, l'ensemble des items et l'ensemble de transactions doivent être représentés par des variables. En effet, une variable Booléenne X_a est associée à chaque item $a \in \Omega$ et une variable Booléenne T_i est associée à chaque transaction $t_i \in \mathcal{D}$. Un itemset I est représenté en fixant $X_a = 1$ pour tout $a \in I$ et $X_a = 0$ pour tout $a \notin I$. Les variables T_t représentent les transactions qui contiennent l'itemset I, c-à-d, $T_t = 1$ si et seulement si $t \in C(I, \mathcal{D})$. Une affectation de toutes les variables X_a (resp. T_t) correspond à un itemset I (resp. la couverture de l'itemset I).

Nous montrons maintenant que la contrainte de couverture peut être formulée comme suit :

$$\forall t \in \{1, \ldots, m\},\ T_t = 1 \leftrightarrow \sum_{a \in \Omega} X_a(1 - \mathcal{D}_{ta}) = 0 \tag{3.1}$$

Notons que $\mathcal{D}_{ta}$ est une constante qui représente l'appartenance de l'item a à la transaction t dans la base $\mathcal{D}$. La contrainte 3.1 peut être réécrite d'une façon équivalente comme suit :

$$\forall t \in \{1, \ldots, m\},\ T_t = 1 \leftrightarrow \bigwedge_{a \in \Omega} (X_a = 0 \vee \mathcal{D}_{ta} = 1)$$

Avec la contrainte de couverture, une variable de transaction T_t n'est vraie que si la transaction correspondante couvre l'itemset I, c-à-d, il n'existe pas un items a qui apparaît dans I ($X_a = 1$) et qui n'apparaît pas dans T_t ($\mathcal{D}_{ta} = 0$). Le comptage du support de l'itemset I peut maintenant être réalisé en comptant le nombre de transactions pour lesquelles $T_t = 1$.

$$\sum_{t \in \{1,\ldots,m\}} T_t \geqslant \alpha \tag{3.2}$$

Nous pouvons maintenant modéliser le problème de fouille d'itemsets fréquents en combinant la contrainte de couverture 3.1 et la contrainte de support minimum 3.2.

Exemple 3.12. *Reprenons la base de données transactionnelles $\mathcal{D}$ présentée dans la table 3.1. Pour un seuil minimum de support $\alpha = 2$, la modélisation du problème de fouille d'itemsets fréquents dans $\mathcal{D}$ en CP correspond au contraintes suivantes :*

$$\begin{aligned}
&T_1 = 1 \leftrightarrow X_e + X_f + X_g = 0\\
&T_2 = 1 \leftrightarrow X_c + X_d + X_g = 0\\
&T_3 = 1 \leftrightarrow X_d + X_e + X_f + X_g = 0\\
&T_4 = 1 \leftrightarrow X_b + X_e + X_g = 0\\
&T_5 = 1 \leftrightarrow X_a + X_b + X_c + X_d + X_e + X_f = 0\\
&T_6 = 1 \leftrightarrow X_a + X_b + X_c + X_e + X_f + X_g = 0\\
&T_7 = 1 \leftrightarrow X_a + X_b + X_c + X_e + X_f = 0\\
&T_1 + T_2 + T_3 + T_4 + T_5 + T_6 + T_7 \geqslant 2
\end{aligned}$$

Pour améliorer la résolution, une autre modélisation équivalente de la contrainte de support minimum a été proposée [RGN08]. L'idée principale est de formuler une contrainte de support sur chaque item individuellement. Cela réduit considérablement la taille de l'arbre de recherche car l'item qui ne vérifie plus sa contrainte de support minimum est propagé à zéro au cours de la recherche. La nouvelle contrainte de support minimum est donnée par l'équation suivante :

$$\forall a \in \Omega,\ X_a = 1 \rightarrow \sum_{t \in \{1,\ldots,m\}} T_t \mathcal{D}_{ta} \geqslant \alpha \tag{3.3}$$

Exemple 3.13. *Reprenons l'exemple précédent. La contrainte $T_1 + T_2 + T_3 + T_4 + T_5 + T_6 + T_7 \geqslant 2$ est remplacée par l'ensemble de contraintes suivant :*

$$\begin{aligned}
&X_a = 1 \leftarrow T_1 + T_2 + T_3 + T_4 \geqslant 2\\
&X_b = 1 \leftarrow T_1 + T_2 + T_3 \geqslant 2\\
&X_c = 1 \leftarrow T_1 + T_3 + T_4 \geqslant 2\\
&X_d = 1 \leftarrow T_1 + T_4 + T_6 + T_7 \geqslant 2\\
&X_e = 1 \leftarrow T_2 \geqslant 2\\
&X_f = 1 \leftarrow T_2 + T_4 \geqslant 2\\
&X_g = 1 \leftarrow T_5 + T_7 \geqslant 2
\end{aligned}$$

Nous pouvons remarquer que la variable X_e est propagée directement à zéro avant de commencer la recherche. Supposons maintenant que nous commençons par affecter $X_c = 1$, alors, les variables X_f et X_g sont propagées à zéros car les variables T_2, T_5 et T_7 sont propagées à zéros par l'affectation de $X_c = 1$.

Pour extraire les itemsets fermés, Guns et al [RGN08] ont modélisé l'intuition suivante : si la couverture d'un itemset I est inclue dans la couverture d'un item a, alors, a doit être dans I pour que I soit fermé. Cela conduit à une contrainte similaire à la contrainte 3.3.

$$\forall a \in \Omega,\ X_a = 1 \leftrightarrow \sum_{t \in \{1,\ldots,m\}} T_t(1 - \mathcal{D}_{ta}) = 0 \tag{3.4}$$

Autrement dit, si un item X_a apparaît dans l'ensemble de transactions couvrant un itemset I, alors, X_a doit être à 1. L'extraction de l'ensemble des itemsets fermés correspond à la conjonction de contraintes de couvertures 3.1, de support minimum 3.4 et de fermeture 3.4.

De la même façon, l'extraction des itemsets maximaux est obtenue en modélisant le fait que si un item a apparaît dans au moins α transactions dans la couverture d'un itemset I, alors, a doit appartenir à I pour que I soit maximal. Cela est obtenu en ajoutant à la contrainte de couverture 3.1, la contrainte suivante :

$$\forall a \in \Omega,\ X_a = 1 \leftrightarrow \sum_{t \in \{1,\ldots,m\}} T_t \mathcal{D}_{ta} \geqslant \alpha \tag{3.5}$$

Exemple 3.14. *Reprenons l'exemple 3.12. La contrainte de fermeture correspond à la conjonction des contraintes suivantes :*

$X_a = 1 \leftrightarrow T_5 + T_6 + T_7 = 0$
$X_b = 1 \leftrightarrow T_4 + T_5 + T_6 + T_7 = 0$
$X_c = 1 \leftrightarrow T_2 + T_5 + T_6 + T_7 = 0$
$X_d = 1 \leftrightarrow T_2 + T_3 + T_5 = 0$
$X_e = 1 \leftrightarrow T_1 + T_3 + T_4 + T_5 + T_6 + T_7 = 0$
$X_f = 1 \leftrightarrow T_1 + T_3 + T_5 + T_6 + T_7 = 0$
$X_g = 1 \leftrightarrow T_1 + T_2 + T_3 + T_4 + T_6 = 0$

Alors que la contrainte de maximalité correspond à la conjonction des contraintes suivantes :

$X_a = 1 \leftrightarrow T_1 + T_2 + T_3 + T_4 \geqslant 2$
$X_b = 1 \leftrightarrow T_1 + T_2 + T_3 \geqslant 2$
$X_c = 1 \leftrightarrow T_1 + T_3 + T_4 \geqslant 2$
$X_d = 1 \leftrightarrow T_1 + T_4 + T_6 + T_7 \geqslant 2$
$X_e = 1 \leftrightarrow T_2 \geqslant 2$
$X_f = 1 \leftrightarrow T_2 + T_4 \geqslant 2$
$X_g = 1 \leftrightarrow T_5 + T_7 \geqslant 2$

D'autres contraintes ont été modélisées en CP [GNDR11] comme la contrainte de taille maximum (resp. minimum) des itemsets qui peut être considérée en rajoutant simplement la contrainte suivante pour un seuil γ :

$$\sum_{\forall a \in \Omega} X_a \leqslant \gamma \ \ (resp. \geqslant \gamma) \tag{3.6}$$

Le passage à l'échelle est un problème majeur de l'approche basé sur la modélisation en CP. En effet, pour résoudre ce problème deux approches principales ont été proposées. La première consiste à proposer des contraintes globales avec des algorithmes de filtrages dédiés [LLL+16, SAG17] et la deuxième consiste à décomposer le problème en sous problèmes plus simples, puis résoudre séquentiellement les instances des sous problèmes [JSS15a]. Nous détaillons seulement l'approche basée sur la décomposition car d'un côté elle est basée sur la modélisation en SAT et d'un autre côté nous l'avons utilisé dans nos contributions. Avant de détailler le principe de décomposition, nous présentons l'encodage SAT du problème de fouille d'itemsets fréquents [JSS13]. De la même façon que l'encodage CP présenté précédemment, la contrainte de couverture est d'abord encodée en SAT, ensuite la contrainte de support minimum est encodée avec une simple contrainte de cardinalité.

Pour représenter un itemset I en SAT, une variable propositionnelle p_a est associée à chaque item $a \in \Omega$ et $I = \{p_a | p_a = 1 \forall a \in \Omega\}$. Pour capturer la couverture de I dans $\mathcal{D}$, une variable propositionnelle q_t est associée à chaque transaction $t \in \{1, \ldots, m\}$. La couverture de I dans $\mathcal{D}$ correspond à l'ensemble $\{q_t | q_t = 1, \forall t \in \{1, \ldots, m\}\}$. Cet ensemble est capturé par la contrainte suivante :

$$\bigwedge_{t \in \{1,\ldots,m\}} (\neg q_t \leftrightarrow \bigvee_{a \in \Omega \setminus I_t} p_a) \tag{3.7}$$

La formule 3.7 exprime que l'itemset candidat n'est pas supporté par la $t^{ème}$ transaction (q_t est faux), quand il existe un item a ($p_a = 1$) qui n'appartient pas à la transaction t ($a \in \Omega \setminus I_t$). En conséquence, $q_t = 1$ si et seulement si l'itemset candidat est supporté par la $t^{ème}$ transaction.
Pour un seuil minimum de support α, la contrainte de support est exprimée par la contrainte de cardinalité suivante :

$$\sum_{t \in \{1,\ldots,m\}} q_t \geqslant \alpha \tag{3.8}$$

La conjonction des deux formules 3.7 et 3.8 correspond exactement au problème de fouille d'itemsets fréquents et chaque modèle de cette formule correspond à un itemset fréquent unique.

Même si la taille de l'encodage est polynomiale par rapport à la taille de la base de données transactionnelle d'entrée, cela ne signifie pas qu'elle est raisonnable. En effet, pour une base de donnée avec 1000 items et 50000 transactions, la taille de la formule qui encode la couverture est de l'ordre de 50 millions de clauses. Cette taille est importante et ingérable pour un solveur SAT. Afin de résoudre ce problème, une approche qui permet de décomposer l'encodage décrit précédemment a été proposée dans [JSS15a].

Étant donné une base de données transactionnelles $\mathcal{D}$ et un itemset S, nous utilisons $\mathcal{D}|_S$ pour noter la base de données transactionnelles $\{(i, I_i) \in \mathcal{D} | I_i \cap S \neq \emptyset\}$ et Σ pour noter l'encodage SAT du problème d'itemsets fréquents. Soit k un nombre entier positif inférieur ou égale à $|\Omega|$. Une k-partition de Ω est une structure de la forme $(\{S_1, \ldots, S_k\}, \prec)$ où $\{S_1, \ldots, S_k\}$ est une partition de Ω en k sous-ensembles et $\prec$ est un ordre sur $\{S_1, \ldots, S_k\}$.

Soit $\mathcal{D}$ une base de données transactionnelles, $\mathcal{P} = (W, \prec)$ une k-partition de Ω, $S \in W$ et α un seuil minimum de support. Nous utilisons $\Sigma(\mathcal{D}, \alpha, S, \mathcal{P})$ pour noter la formule propositionnelle suivante :

$$\Sigma(\mathcal{D}|_S, \alpha) \wedge (\bigvee_{a \in S} p_a) \wedge (\bigwedge_{b \in \cup_{S' \prec S} S'} \neg p_b) \tag{3.9}$$

Dans la formule 3.9, la clause $(\bigvee_{a \in S} p_a)$ exprime que les itemsets doivent contenir au moins un item de S. Par contre, la sous formule $(\bigwedge_{b \in \cup_{S' \prec S} S'} \neg p_b)$ nous permet d'éviter la régénération des itemsets précédemment générés en utilisant les éléments précédents de la partition $S' \prec S$. La proposition suivante montre que l'approche par décomposition permet l'obtention de tous les itemsets fréquents.

Proposition 3.1. *Soit $\mathcal{D}$ une base de données transactionnelles, $\mathcal{P} = (W, \prec)$ une k-partition de Ω, $S \in W$ et α un seuil minimum de support. Alors, $\mathcal{I}$ est un modèle de $\Sigma(\mathcal{D}, \alpha, S, \mathcal{P})$ si et seulement si les propriétés suivantes sont satisfaites :*

1. $I = \{a \in \Omega \mid \mathcal{I}(p_a) = 1\} \in FI(\mathcal{D}, \alpha)$;
2. $\mathcal{C}(I, \mathcal{D}) = \{t \in \mathbb{N} \mid \mathcal{I}(q_t) = 1\}$;
3. $I \cap S \neq \emptyset$; *et*
4. $I \cap \bigcup_{S' \prec S} S' = \emptyset$.

La proposition suivante montre que l'approche de décomposition proposée permet d'éviter la redondance.

Proposition 3.2. *Soit $\mathcal{D}$ une base de données transactionnelles, $\mathcal{P} = (W, \prec)$ une k-partition de Ω, $S, S' \in W$ tel que $S \neq S'$ et α un seuil minimum de support. Alors $\mathcal{M}(\Sigma(\mathcal{D}, \alpha, S, \mathcal{P})) \cap \mathcal{M}(\Sigma(\mathcal{D}, \alpha, S', \mathcal{P})) = \emptyset$.*

$\mathcal{M}(\Sigma)$ représente l'ensemble de modèles de la formule Σ.

L'algorithme 15 décrit la procédure simple utilisant la méthode basée sur les partitions pour énumérer tous les itemsets fréquents proposée dans [JSS15a].

Algorithm 15: Algorithme $SAT^{\mathcal{P}}$

Input: Une base de données transactionnelles $\mathcal{D}$, un seuil α et une k-partition $\mathcal{P} = (\{S_1, \ldots, S_k\}, \prec)$
Output: R : Ensemble des itemsets fréquents dans $\mathcal{D}$ par rapport à α

```
1 R = ∅ ;
2 for (i = 1 à k ) do
      /* S_1 ≺ ··· ≺ S_k                                   */
3     R ← R ∪ enumModels(Σ(D, α, S_i, P));
4 return R;
```

Il est intéressant de noter qu'en utilisant la proposition 3.2, nous savons qu'il n'y a pas d'itemset qui va être généré à partir de deux itérations différentes dans la boucle for. La procédure $enumModels(\mathcal{F})$ énumère tous les modèles de la formule propositionnelle $\mathcal{F}$.

Notons aussi qu'une simple partition $\mathcal{P}$ a été proposée dans [JSS15a] en considérant que chaque itemset S correspond à un seul item et que $S' \prec S$ si le support de S' est inférieur au support de S dans $\mathcal{D}$. Cela permet de réduire largement la taille de la formule propositionnelle et augmente l'efficacité des solveurs en accélérant la propagation unitaire.

3.3.2 Extraction de l'ensemble des k-motifs

Le problème d'extraction sous contraintes de tous les k-motifs (ou k-itemsets) est une généralisation du problème de fouille d'itemsets fréquents qui a été proposée par Guns et al dans [GNDR13]. Cette généralisation permet d'engendrer un ensemble de problèmes liés à la fouille des données transactionnelles comme l'extraction des règles d'association ou le clustering conceptuel. En effet, le problème d'extraction des k-itemsets en utilisant CP fournit une approche plus générale de fouille de données qui peut être instanciée à une grande variété de tâches. Cet vision générale permet d'éviter le processus d'extraction à plusieurs étapes par exemple pour fouiller les règles d'association avec une approche dédiée, il faut d'abord extraire les itemsets fréquents puis garder les paires qui vérifient la contrainte de confiance. Alors qu'en modélisant ce problème en un problème de fouille de 2-itemsets sous les contraintes de support et de confiance, on peut extraire les règles d'association en une seul étapes. L'idée principale est de récupérer la couverture de chacun des k itemsets dans une base de données transactionnelles $\mathcal{D}$. Ensuite, exprimer toutes les contraintes locales nécessaires sur chaque itemset et globales entres les k itemsets.

Pour résoudre le problème d'extraction de l'ensemble des k-itemsets sous contraintes, un cadre général basé sur la programmation par contraintes a été proposé dans [GNDR13]. Étant donné une base de données transactionnelles $\mathcal{D}$ contenant m transactions, construite sur un ensemble d'items Ω de taille n. L'objectif est de trouver tous les ensembles $\{P_1, \ldots, P_k\}$ qui vérifient un ensemble de contraintes locales sur chaque itemset P_i et un ensemble de contraintes globales entres les itemsets P_i. Pour modéliser ce problème, des variables Booléennes X_a^i sont associées à chaque item $a \in \Omega$ pour représenter l'appartenance de l'item a à l'itemset P_i avec $i \in \{1, \ldots, k\}$, c-à-d, $P_i = \{X_a^i = 1,\ \forall a \in \Omega\}$. De plus, des variables T_t^i sont associées à chaque transaction $t \in \{1, \ldots, m\}$ et $i \in \{1, \ldots, k\}$. La couverture de l'itemset P_i est donnée par l'ensemble suivant : $\{T_t^i = 1,\ \forall t \in \{1, \ldots, m\}\}$. De la même façon que dans la sous section précédente, les contraintes locales peuvent être exprimées en utilisant les variables liées à chaque itemset. Par exemple la contrainte de taille minimum γ sur tous les k itemsets correspond à la contrainte suivante :

$$\forall i \in \{1, \ldots, k\},\ \sum_{a \in \Omega} X_a^i \leqslant \gamma \tag{3.10}$$

Dans [GNDR13], de nombreux problèmes ont été instanciés en tant que problèmes de fouille de k-itemsets. Par exemple, le problème de fouille des règles d'association peut être vu comme un problème d'extraction des 2-itemsets (ou des paires (P_1, P_2)) [KBC11] qui vérifient les contraintes de support et de confiance. En encodant la contrainte de couverture 3.1 sur les variables de P_1 et celles de P_2 nous auront la couverture de chacun . Pour capturer la couverture de l'union $P_1 \cup P_2$ il suffit d'ajouter m variables de transaction $T_t{}'$ et exprimer l'intersection avec la contrainte suivante :

$$\forall t \in \{1, \ldots, m\},\ (T_t{}' = 1) \leftrightarrow (T_t^1 = 1) \wedge (T_t^2 = 1) \tag{3.11}$$

Maintenant il suffit d'exprimer la contrainte de support minimum par rapport à un seuil α en utilisant la couverture de l'itemset $P_1 \cup P_2$ comme suit :

$$\sum_{t \in \{1,\ldots,m\}} T_t{}' \geqslant \alpha \tag{3.12}$$

La contrainte de confiance minimum par rapport à un seuil β est exprimée en utilisant la couverture de P_1 et la couverture de $P_1 \cup P_2$ par la contrainte suivante :

$$\Big(\sum_{t \in \{1,\ldots,m\}} T_t{}'\Big) - \beta\Big(\sum_{t \in \{1,\ldots,m\}} T_t^1\Big) \geqslant 0 \tag{3.13}$$

Cette instanciation a été présentée dans [KBC11] avec deux autres instanciations de deux problèmes de fouille de règles d'association, à savoir la fouille des règles d'association inattendues et la fouille des règles d'exception. Il est à noter qu'aucune comparaison avec les approches dédiées n'est faite et aucune modélisation des autres types de règles d'association n'a été proposée. Pour plus de détails sur d'autres instanciations de problèmes, nous référons le lecteur vers l'article de Guns et al [GNDR13]

3.4 Conclusion

Ce chapitre est une introduction au problème de fouille de règles d'association qui est en lien direct avec nos contributions présentées dans le chapitre 5. En effet, nous avons d'abord commencé par la présentation du problème de fouille d'itemsets qui constitue l'étape de base pour l'extraction des règles d'association. Puis, une description de quelques approches spécialisées dans la fouille des itemsets fréquents avec leurs représentation condensées a été donnée. Ensuite, le problème de fouille de règles d'association est défini et plusieurs variantes importantes sont présentées. Enfin, un aperçu sur les modélisations du problème de fouille d'itemsets en utilisant CP et SAT est fourni.

Deuxième partie

Contributions

Chapitre 4

Clustering de données complexes représentées par des formules logiques

Sommaire

Le clustering a été étudié de manière approfondie pour traiter les différents types de données. Ces données sont généralement représentées par des vecteurs d'attributs à n dimensions décrits par des valeurs numériques ou catégorielles. Les données symboliques sont un autre concept où les objets sont plus complexes tels que les intervalles, multi-catégoriels ou multi-modale. Cependant, de nouvelles applications pourraient donner lieu à des données encore plus complexes décrivant, par exemple, les désirs, les contraintes et les préférences des agents recueillis de différentes manières en utilisant par exemples des questionnaires. A titre d'exemple, on peut citer les systèmes de configuration généralement conçus pour fournir des produits personnalisés répondant aux différentes exigences du client. Ces exigences sont généralement modélisées par des contraintes ou des formules logiques (e.g. [HFS+14]). Ces données sur les exigences des clients ou les modèles de données fournis par les systèmes de configuration sont les types de données complexes qui nous intéressent. Elles peuvent être représentées par des formules logiques (exigences) ou par des modèles (les produits répondant aux exigences). Les données peuvent également représenter des entités plus complexes telles que les bases de données transactionnelles. En effet, supposons que nous avons collecté plusieurs bases de données transactionnelles d'une chaîne de magasins vendant les mêmes produits, on peut s'intéresser à la recherche des magasins similaires (clusters) ou des magasins ayant le même comportement. Ceci pourrait aider le gestionnaire de la chaîne à mieux définir sa politique commerciale. Dans les deux exemples précédents, les données peuvent être mieux représentées comme un ensemble de formules propositionnelles ou comme des ensembles de modèles.

Dans ce chapitre, nous introduisons un nouveau cadre de clustering, où les objets complexes sont décrits par des formules propositionnelles. Nous étendons d'abord les deux méthodes les plus connues à

Clients	c_1	c_2	c_3	c_4
$\mathcal{F}_{c_i}$	$\neg r$	d	$g \wedge r$	p
$\mathcal{M}(\mathcal{F}_{c_i} \wedge \mu)$	010101	011001		010101
	010110	011010		011001
	011001	101001	100101	100101
	011010	101010	100110	101001

TABLE 4.1 – Représentation basée sur la logique des préférences des clients

savoir l'algorithme de partitionnement k-$means$ et le clustering hiérarchique ascendant pour traiter des objets complexes représentés sous forme de formules propositionnelles. Ensuite, nous introduisons un nouvel algorithme hiérarchique descendant pour regrouper les objets représentés explicitement par des ensembles de modèles. Puis, nous proposons un encodage basé sur la satisfiabilité propositionnelle du problème du clustering des formules propositionnelles sans avoir besoin d'une représentation explicite de leurs modèles. Enfin, des résultats expérimentaux préliminaires qui valident le cadre que nous proposons sont fournis.

4.1 Motivations

Pour motiver le cadre que nous proposons, considérons un exemple simple d'un concessionnaire automobile vendant différentes marques de voitures avec plusieurs options possibles. Pour chaque marque de voiture, plusieurs couleurs et types de carburants sont disponibles. Le concessionnaire automobile a recueilli les préférences de quatre clients. Le premier client ne veut pas de voitures rouges. Le second veut une voiture avec un carburant diesel, tandis que le troisième veut une voiture rouge qui roule à l'essence. Enfin, le quatrième client préfère les voitures de marque Peugeot. En plus de ces préférences des clients, nous considérons également les contraintes d'exclusion mutuelle (mutex), permettant d'exprimer que chaque voiture doit avoir une seule couleur, un seul type de carburant et elle appartient à une seule marque de voiture.

Pour exprimer les différents désirs des clients en utilisant la logique propositionnelle, nous considérons les variables propositionnelles suivantes : r (resp. b) représente la couleur rouge (resp. noir), p (resp. c) représente la marque de voiture Peugeot (resp. Citroen) et d (resp. g) représente les voitures utilisant comme carburant le diesel (ou essence).

Les contraintes d'exclusion mutuelle sont exprimées par la formule suivante :

$$\mu = [(r \wedge \neg b) \vee (b \wedge \neg r)] \wedge [(g \wedge \neg d) \vee (d \wedge \neg g)] \wedge [(p \wedge \neg c) \vee (c \wedge \neg p)] \tag{4.1}$$

Dans la table 4.1, nous associons une formule propositionnelle $\mathcal{F}_{c_i}$ exprimant les désirs de chaque client c_i. Nous fournissons également l'ensemble des modèles satisfaisant à la fois les désirs du client et les contraintes d'exclusion mutuelle ($\mathcal{M}(\mathcal{F}_{c_i} \wedge \mu)$). La présentation des modèles suit l'ordre des variables : $r \prec b \prec d \prec g \prec c \prec p$. Dans la figure 4.1, nous donnons une représentation graphique des préférences des quatre clients.

Cet exemple illustratif montre la puissance d'expression d'une représentation de données basée sur la logique propositionnelle qui donne la possibilité de définir à la fois des contraintes d'utilisateurs et des

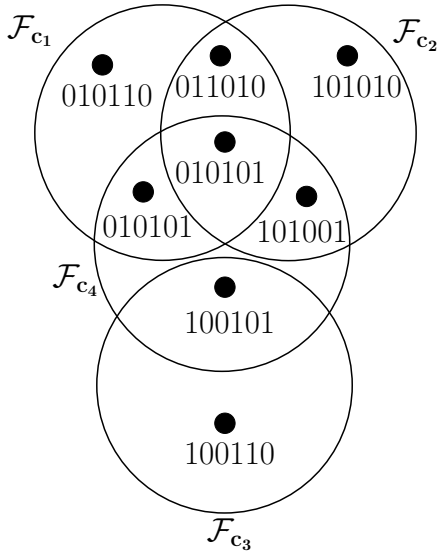


FIGURE 4.1 – Représentation graphique des préférences des clients

contraintes de contexte comme les exclusions mutuelles montrées dans l'exemple tout en assurant une représentation compacte de l'ensemble de modèles qui peut être très grand.

4.2 Adaptation des algorithmes de clustering

Dans cette section, nous présentons notre extension de certains algorithmes de clustering, à savoir l'algorithme hiérarchique ascendant et le k-$means$ pour prendre en compte les objets représentés par des formules de la logique propositionnelle.

Commençons par quelques notations et définitions nécessaires. Nous notons d'abord $\mathcal{P}(k, \Phi)$ le problème du clustering d'un ensemble de formules propositionnelles $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ en un ensemble de k clusters de formules avec $k \leqslant n$.

Définition 4.1 (Solution d'un problème de clustering de formules propositionnelles)**.** *Soit $\mathcal{C}$ une partition sur Φ. $\mathcal{C}$ est une solution de $\mathcal{P}(k, \Phi)$ si et seulement si :*

1. $|\mathcal{C}| = k$.
2. $\bigcup_{\mathcal{C}_i \in \mathcal{C}} \mathcal{C}_i = \Phi$ *avec* $C_i \cap C_j = \emptyset$ *pour* $i \neq j$.
3. $\forall \mathcal{C}_i \in \mathcal{C}, \mathcal{M}(\bigwedge_{\mathcal{F} \in \mathcal{C}_i} \mathcal{F}) \neq \emptyset$.

La définition 4.1 fixe les trois conditions nécessaires pour qu'une partition de formules propositionnelles $\mathcal{C}$ soit acceptée pour un problème de clustering de formules propositionnelles $\mathcal{P}(k, \Phi)$. La première condition permet d'avoir une partition de taille égale au nombre de clusters k fixé par l'utilisateur. La deuxième condition permet d'avoir un clustering complet en partitionnant toutes les formules en k clusters et que chaque formule appartient à une seule partition sans aucun chevauchement entre les partitions. La dernière condition oblige les formules de chaque partition $\mathcal{C}_i$ à partager au moins un modèle pour assurer la cohérence dans chaque cluster.

Définition 4.2 (Cohérence d'un problème de clustering de formules propositionnelles)**.** *Un problème de clustering $\mathcal{P}(k, \Phi)$ est cohérent si et seulement s'il admet au moins une solution.*

Il est important de noter que la cohérence de la solution de clustering force les formules d'un même cluster à partager au moins un modèle. Cette propriété, permet à un concessionnaire dans l'exemple précédent, de limiter par exemple les choix sur les modèles de voitures à l'union des modèles partagés dans chaque cluster. Ceci, garantit la satisfaction de tous les clients.

4.2.1 k-$means$ pour le clustering de formules propositionnelles

Étant donné un ensemble de n points de données dans un espace à d dimensions $\mathbb{R}^d$ et un nombre entier k positif, l'algorithme k-$means$ détermine un ensemble de k points dans $\mathbb{R}^d$, appelés centroïdes, de manière à minimiser une fonction objectif égale à la somme des distances entre chaque point et son centroïde le plus proche. Pour étendre l'algorithme k-$means$ au clustering d'objets décrits par des formules propositionnelles, nous devons définir :

1. une mesure de distance entre deux formules propositionnelles ;
2. un centroïde qui représente un cluster donné ;
3. une fonction objectif à optimisée.

Dans la suite, nous montrons comment l'algorithme k-$means$ peut être adapté en reformulant les trois composantes précédentes.

Rappelons qu'une formule propositionnelle $\mathcal{F}$ peut être exprimée de manière équivalente par son ensemble de modèles $\mathcal{M}(\mathcal{F})$. Avec cette représentation en tête, on peut considérer que deux formules $\mathcal{F}_1$ et $\mathcal{F}_2$ sont similaires si leur ensemble de modèles communs $\mathcal{M}(\mathcal{F}_1) \cap \mathcal{M}(\mathcal{F}_2)$ est plus grand que l'ensemble des autres modèles (distinctifs) $\mathcal{M}(\mathcal{F}_1) \setminus \mathcal{M}(\mathcal{F}_2) \cup \mathcal{M}(\mathcal{F}_2) \setminus \mathcal{M}(\mathcal{F}_1)$. Ce type de similarité est lié au modèle de contraste très connu de la similarité proposé dans un article de Tversky [Tve77]. En effet, le modèle de contraste propose que deux objets sont très similaires s'ils ont plus de caractéristiques communes et moins de caractéristiques distinctives. Ils sont moins similaires s'ils ont plus de caractéristiques distinctives et moins de caractéristiques communes. Dans la définition suivante, nous décrivons la version normalisée du modèle de similarité de contraste, appelé le modèle de ratio [Tve04].

Définition 4.3. *Soient a et b deux objets décrits par deux ensembles de caractéristiques A et B respectivement. La similarité entre a et b est définie comme suit :*

$$s\left(a,b\right) = \frac{f\left(A \cap B\right)}{f\left(A \cap B\right) + \alpha f\left(A \setminus B\right) + \beta f\left(B \setminus A\right)}$$

$\alpha, \beta \geqslant 0$

Les coefficients positifs α et β reflètent les poids attribués aux caractéristiques distinctives des deux objets a et b. Nous supposons généralement que f est une fonction de correspondance satisfaisant la propriété d'additivité $f(A \cup B) = f(A) + f(B)$, à chaque fois que A et B sont disjoints. Le modèle de ratio définit une valeur normalisée de similarité telle que $0 \leqslant s(a,b) \leqslant 1$.

Le modèle de similarité de contraste est particulièrement adapté dans notre contexte. Pour étendre la définition 4.3, nous considérons la relation entre les opérations ensemblistes et les opérateurs logiques. En effet, à partir de la perspective sémantique, l'union (resp. intersection) entre les ensembles est la disjonction (resp. conjonction) entre les formules propositionnelles. La différence entre les ensembles peut être exprimée en utilisant à la fois des opérateurs de conjonction et de négation, tandis que la différence symétrique entre les ensembles peut être exprimée en utilisant l'opérateur logique xor ($\oplus$). En effet, nous avons :

$$\mathcal{M}(\mathcal{F}_1) \setminus \mathcal{M}(\mathcal{F}_2) \cup \mathcal{M}(\mathcal{F}_2) \setminus \mathcal{M}(\mathcal{F}_1) = \mathcal{M}((\mathcal{F}_1 \wedge \neg\mathcal{F}_2) \vee (\mathcal{F}_2 \wedge \neg\mathcal{F}_1)) = \mathcal{M}(\mathcal{F}_1 \oplus \mathcal{F}_2)$$

En utilisant ces relations, nous dérivons l'extension suivante du modèle de ratio.

Définition 4.4. *Soient a et b deux objets décrits par deux formules propositionnelles $\mathcal{F}_1$ et $\mathcal{F}_2$ respectivement. La similarité entre a et b est définie comme suit :*

$$s(a,b) = \frac{f(\mathcal{F}_1 \wedge \mathcal{F}_2)}{f(\mathcal{F}_1 \wedge \mathcal{F}_2) + \alpha f(\mathcal{F}_1 \wedge \neg\mathcal{F}_2) + \beta f(\mathcal{F}_2 \wedge \neg\mathcal{F}_1)}$$

$\alpha, \beta \geqslant 0$

Dans notre contexte, nous considérons qu'aucune distinction n'est faite entre la mesure de $\mathcal{F}_1 \wedge \neg\mathcal{F}_2$ et $\mathcal{F}_2 \wedge \neg\mathcal{F}_1$. En effet, nous dérivons la mesure de similarité suivante.

Définition 4.5. *Soient a et b deux objets décrits par deux formules propositionnelles $\mathcal{F}_1$ et $\mathcal{F}_2$ respectivement. La similarité entre a et b est définie comme suit :*

$$s(a,b) = \frac{f(\mathcal{F}_1 \wedge \mathcal{F}_2)}{f(\mathcal{F}_1 \wedge \mathcal{F}_2) + \gamma f(\mathcal{F}_1 \oplus \mathcal{F}_2)}, \gamma \geqslant 0$$

A partir de la définition 4.4 (ou de la définition 4.5), en instanciant $\alpha = \beta = 1$ (resp. $\gamma = 1$), nous dérivons une variante logique du fameux coefficient de similarité de Jaccard (resp. distance) [Jac12].

Définition 4.6. *Soient a et b deux objets décrits par deux formules propositionnelles $\mathcal{F}_1$ et $\mathcal{F}_2$ respectivement. La similarité et la distance entre a et b ou entre $\mathcal{F}_1$ et $\mathcal{F}_2$ sont respectivement définies comme suit :*

$$s_J(a,b) = s_J(\mathcal{F}_1, \mathcal{F}_2) = \frac{f(\mathcal{F}_1 \wedge \mathcal{F}_2)}{f(\mathcal{F}_1 \vee \mathcal{F}_2)}$$
$$d_J(a,b) = 1 - s_J(a,b) = d_J(\mathcal{F}_1, \mathcal{F}_2)$$

Comme mentionné précédemment, en considérant la représentation basée sur les ensembles de modèles des formules propositionnelles, nous définissons la fonction f comme :

$$f : \left| \begin{array}{ccc} F_{\mathcal{P}} & \longrightarrow & \mathbf{N} \\ \mathcal{F} & \longmapsto & |\mathcal{M}(\mathcal{F})| \end{array} \right.$$

La fonction f vérifie la propriété d'additivité. Nous avons $\mathcal{M}(\mathcal{F}_1 \vee \mathcal{F}_2) = \mathcal{M}(\mathcal{F}_1) \cup \mathcal{M}(\mathcal{F}_2)$.

Le calcul de f implique la résolution d'un problème de comptage de modèles $\#P$-Complet. Comme nous l'avons vu dans le premier chapitre, plusieurs outils de comptage de modèles exacts ou approximatifs efficaces ont été conçus ces dernières années (e.g.[GHSS07, CMV13]) [1].

Exemple 4.1. *Reprenons l'exemple des préférences des clients décrites dans la table 4.1. En utilisant d_J, nous obtenons la matrice de distance représentée dans la table 4.2.*

Selon la mesure de similarité et de distance présentée dans la définition 4.6, l'objectif est de regrouper les objets représentés par des formules propositionnelles en groupes (clusters) de formules qui ont plus de modèles en communs et moins de modèles distinctifs.

Définissons maintenant le représentant d'un groupe de formules propositionnelles.

1. http://tools.computational-logic.org/content/sharpCDCL.php
http://www.cs.cornell.edu/~sabhar/$#$software

	$\mathcal{F}_{c_1}$	$\mathcal{F}_{c_2}$	$\mathcal{F}_{c_3}$	$\mathcal{F}_{c_4}$
$\mathcal{F}_{c_1}$	0	$\frac{2}{3}$	1	$\frac{2}{3}$
$\mathcal{F}_{c_2}$	$\frac{2}{3}$	0	1	$\frac{2}{3}$
$\mathcal{F}_{c_3}$	1	1	0	$\frac{4}{5}$
$\mathcal{F}_{c_4}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{4}{5}$	0

TABLE 4.2 – Matrice de distance de l'exemple du concessionnaire automobile (Tableau 4.1)

Définition 4.7. *Soit $\mathcal{C}_i$ un cluster contenant n_i formules $\{\mathcal{F}_{1_i}, \mathcal{F}_{2_i}, \ldots, \mathcal{F}_{n_i}\}$. Nous définissons le représentant du cluster (également appelé centroïde) $\mathcal{O}_{\mathcal{C}_i}$ du cluster $\mathcal{C}_i$ comme suit :*

$$\mathcal{O}_{\mathcal{C}_i} = \mathcal{F}_{1_i} \wedge \mathcal{F}_{2_i} \wedge \ldots \wedge \mathcal{F}_{n_i}$$

Il est important de noter que dans notre extension, l'objectif est de regrouper les formules en groupes cohérents. Par conséquent, la formule représentant un cluster donné doit être cohérente.

Maintenant, nous définissons la fonction objectif de notre clustering. La définition 4.8 introduit la fonction objectif du k-$means$ classique.

Définition 4.8. *Soit $\mathcal{P}(k, \Phi)$ un problème du clustering d'un ensemble de formules propositionnelles $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ en $k \leqslant n$ clusters $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$. La fonction objectif est définie à l'aide du critère d'erreur absolue (AEC) comme suit :*

$$C^* = \arg\min_{\mathcal{C}} \sum_{i=1}^{k} \sum_{\mathcal{F} \in \mathcal{C}_i} d_J(\mathcal{F}, \mathcal{O}_{\mathcal{C}_i}) \tag{4.2}$$

Notre algorithme de clustering d'un ensemble de formules propositionnelles peut maintenant être dérivé de l'algorithme k-$means$ classique en utilisant les nouvelles composantes (distance, centroïde et fonction objectif) définies auparavant. Comme mentionné précédemment, nous recherchons un clustering cohérent. Pour satisfaire cette propriété, une formule est ajoutée à un cluster donné, uniquement si elle est cohérente avec son centroïde c'est-à-dire que les formules dans tout cluster doivent partager au moins un modèle. Si cette condition n'est pas remplie, l'algorithme renvoie l'ensemble actuel de clusters s'il existe ou aucun clustering sinon.

Comme le k-$means$ est un algorithme heuristique, il n'y a aucune garantie qu'il convergera vers un optimum global et le résultat dépendra fortement des clusters initiaux ([BMvL12, KV06]). De même, la sélection initiale des k formules à mettre dans chaque cluster doit avoir un impact important sur la stabilité et l'efficacité de la tâche de clustering. Pour illustrer un tel cas, reconsidérons l'exemple donné de la table 4.1. Supposons que $k = 2$, $\mathcal{F}_{c_1}$ et $\mathcal{F}_{c_2}$ sont les deux formules représentatives sélectionnées au hasard. Dans ce cas particulier, $\mathcal{F}_{c_3}$ ne peut être affectée à aucun des deux cluster $\mathcal{C}_1$ et $\mathcal{C}_2$. En effet, le client c_3 préfère une voiture rouge à essence. Ceci est incohérent avec la préférence du client c_1 qui ne veut pas de voiture rouge. De même, le client c_3 ne peut pas être mis dans le cluster $\mathcal{C}_2$ car les clients c_2 et c_3 souhaitent des voitures avec des carburants différents.

Dans notre contexte, une stratégie possible pour surmonter ce cas problématique consiste à initialiser les clusters avec des formules éloignées, idéalement des formules ne partageant pas de modèles. Pour

ce faire, nous commençons par choisir aléatoirement une formule unique, puis la deuxième formule sera celle avec la distance maximale par rapport à la première, et la $i^{ème}$ formule est celle qui maximise la distance minimale entre chacune des formules restantes et celles déjà choisies. En utilisant cette heuristique d'initialisation, le cas particulier identifié dans l'exemple du concessionnaire automobile est évité. En effet, supposons que l'algorithme commence par choisir la formule $\mathcal{F}_{c_1}$, la formule suivante représentant le second cluster sera $\mathcal{F}_{c_3}$. Cette heuristique d'initialisation fournit deux clusters cohérents. Cependant, ces cas problématiques ne sont généralement pas évités par cette stratégie heuristique améliorée.

4.2.2 Algorithme hiérarchique ascendant pour le clustering de formules propositionnelles

Les algorithmes hiérarchiques peuvent se comporter mieux que le k-$means$. L'idée principale d'un algorithme hiérarchique ascendant est de construire un dendrogramme tel qu'à chaque niveau les deux clusters les plus proches sont fusionnés. En appliquant un algorithme hiérarchique, nous ferons en sorte que les objets les plus proches les uns des autres, seront nécessairement dans le même cluster. Dans cette adaptation, la similarité entre deux clusters est identique à la similarité entre leurs représentants. De même que pour la définition 4.8, la conjonction de toutes les formules dans un cluster représente son centroïde. Pour fusionner des clusters, nous combinons les deux clusters avec la plus petite distance entre ses centroïdes.

Dans l'algorithme 16, nous présentons notre adaptation de l'algorithme de clustering hiérarchique ascendant pour les formules propositionnelles.

Algorithm 16: Algorithme hiérarchique ascendant pour le clustering des formules propositionnelles

Input: Un ensemble de formules propositionnelles $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$.
Output: Un arbre (*dendogramme*) ou une forêt $\mathcal{T}$

1 **for** $i \leftarrow 1$ *à* n **do**
2 $\quad \mathcal{C}_i \leftarrow \{\mathcal{F}_i\}$;
3 $\quad \mathcal{O}_{\mathcal{C}_i} \leftarrow \mathcal{F}_i$;
4 $\mathcal{T} = (\mathcal{N}, \mathcal{A}) \leftarrow (\{n_i | \mathcal{C}_i \in \mathcal{C}\}, \emptyset)$ ▷ n_i nœud associé à C_i;
5 **while** $(|\mathcal{C}| > 1)$ **do**
6 $\quad (\mathcal{C}_i^*, \mathcal{C}_j^*) \leftarrow \underset{\mathcal{C}_i, \mathcal{C}_j \in \mathcal{C}}{\arg\min}(d_J(\mathcal{O}_{\mathcal{C}_i}, \mathcal{O}_{\mathcal{C}_j}))$;
7 $\quad n_{ij} = \mathcal{C}_{ij} \leftarrow \{\mathcal{C}_i^* \cup \mathcal{C}_j^*\}$ ▷ n_{ij} nœud associé à $\mathcal{C}_{ij}$;
8 $\quad \mathcal{O}_{ij} \leftarrow \mathcal{O}_{\mathcal{C}_i^*} \wedge \mathcal{O}_{\mathcal{C}_j^*}$;
9 $\quad$ **if** $(\mathcal{O}_{ij} \vDash \bot)$ **then**
10 $\quad\quad$ **return** $\mathcal{T}$
11 $\quad$ **else**
12 $\quad\quad \mathcal{C} \leftarrow (\mathcal{C} \setminus \{\mathcal{C}_i^*, \mathcal{C}_j^*\}) \cup \{\mathcal{C}_i^* \cup \mathcal{C}_j^*\}$;
13 $\quad\quad (\mathcal{N}, \mathcal{A}) \leftarrow (\mathcal{N} \cup \{n_{ij}\}, \mathcal{A} \cup \{(n_{ij}, n_i), (n_{ij}, n_j)\})$;
14 **return** $\mathcal{T}$

L'algorithme prend comme entrée un ensemble de formules propositionnelles Φ et renvoie un arbre appelé dendrogramme (ligne 14) ou une forêt lorsqu'une incohérence est rencontrée (ligne 10). Il commence par affecter chaque formule dans un cluster unitaire (lignes 1-3). Dans l'arbre initial, l'ensemble des nœuds correspond aux clusters, tandis que l'ensemble des arcs est un ensemble vide (ligne 5). À

chaque itération, deux clusters avec la distance minimale sont sélectionnés (ligne 6) et fusionnés dans un nouveau cluster (lignes 7-8). Si cette opération de fusion conduit à un nouveau cluster incohérent, l'algorithme s'arrête et renvoie un dendrogramme partiel ou une forêt (ligne 10). Sinon, l'ensemble de clusters avec le dendrogramme associé sont mis à jour (lignes 12-13). L'algorithme renvoie un arbre (ligne 14) lorsque tous les clusters sont fusionnés dans un seul cluster cohérent.

On peut remarquer qu'à la ligne 6, un appel à un oracle # SAT est effectué, conduisant à un nombre global d'appels de $\mathcal{O}(n^2)$. Cependant, l'appel à un oracle SAT (ligne 9) peut être évité, car $\mathcal{O}_{ij}$ n'est incohérent que si $d_J(\mathcal{O}_{\mathcal{C}_i^*}, \mathcal{O}_{\mathcal{C}_j^*}) = 1$ (déjà calculé à la ligne 6). Cependant, lorsque les formules sont représentées par leurs modèles, aucun appel d'oracle n'est nécessaire. En effet, la distance d_J (ligne 6), peut être calculée en utilisant des opérations ensemblistes.

Exemple 4.2. *Pour illustrer le comportement de l'algorithme 16, prenons l'exemple des clients représentés dans la table 4.1. Selon la matrice de distances présentée dans la table 4.2, le dendrogramme représenté sur la figure 4.2 montre une hiérarchie des formules. À chaque niveau (axe des y), nous mentionnons la distance entre les deux clusters fusionnés. Pour obtenir un clustering, nous devons couper le dendrogramme à un niveau donné en fonction du nombre de clusters désiré. Comme le montre la figure 4.2 au niveau 3 du dendrogramme, la fusion de $\mathcal{F}_{c_3}$ avec $\mathcal{F}_{c_1} \wedge \mathcal{F}_{c_2} \wedge \mathcal{F}_{c_4}$ conduit à une incohérence.*

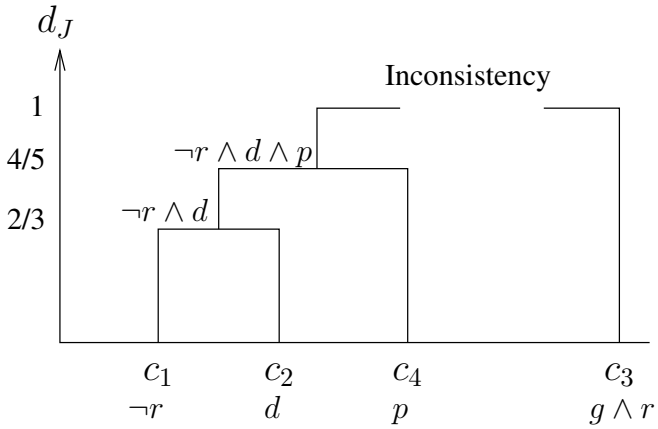


FIGURE 4.2 – Clustering hiérarchique ascendant sur l'exemple du concessionnaire automobile

L'exemple 4.2 montre que le clustering avec un algorithme hiérarchique ascendant peut échouer à résoudre le problème de clustering des formules propositionnelles lorsque la propriété de cohérence est prise en compte. Plus précisément, cela se produit lorsqu'une incohérence est détectée à un niveau donné ce qui empêche l'algorithme d'atteindre le clustering désiré s'il existe.

Supposons que nous voulons regrouper les formules représentées sur la figure 4.3 en deux groupes cohérents. Cet algorithme ne peut pas fournir de solution. En effet, en commençant par fusionner $\mathcal{F}_3$ et $\mathcal{F}_4$, l'algorithme ne peut pas atteindre le cas de deux clusters car d'une part $\mathcal{F}_3 \wedge \mathcal{F}_4$ ne partagent pas de modèles avec toutes les autres formules restantes et d'autre part $\mathcal{F}_5$ et $\mathcal{F}_6$ ne partagent pas de modèles avec $\mathcal{F}_2$ et $\mathcal{F}_1$. Par conséquent, comme les clusters doivent être cohérents, le clustering conduit à au moins trois clusters.

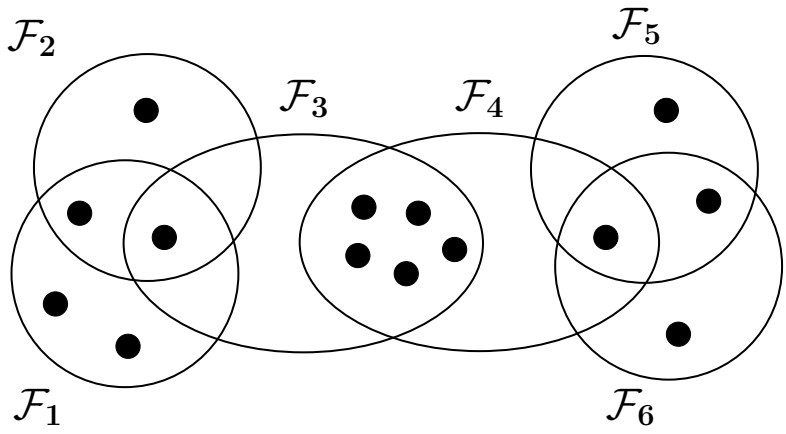


FIGURE 4.3 – Cas problématique pour l'algorithme hiérarchique ascendant

4.3 Algorithme hiérarchique descendant pour une représentation basée sur les modèles

Comme mentionné précédemment, lorsque nous considérons le problème de clustering d'un ensemble de formules propositionnelles $\Phi = \{\mathcal{F}_1, \mathcal{F}_1, \ldots, \mathcal{F}_n\}$ qui ne partagent pas de modèles c'est-à-dire $\Phi \vdash \bot$, l'algorithme hiérarchique ascendant et le $k\text{-}means$ peuvent échouer à trouver un clustering avec le nombre de clusters souhaité. Dans la suite, nous proposons un algorithme hiérarchique descendant (ou divisif) pour partitionner un ensemble de formules propositionnelles représentées par leurs modèles. Contrairement aux algorithmes adaptés, l'algorithme que nous proposons trouve toujours une solution pour le problème de clustering de formules si elle existe.

La variante que nous proposons utilise le problème de *transversal* minimum, que nous rappelons dans les définitions suivantes.

Définition 4.9 (*Transversal*)**.** *H est un transversal d'un ensemble d'ensembles Ω si $\forall S \in \Omega$, $H \cap S \neq \emptyset$.*
Un transversal H est irréductible s'il n'y a pas d'autre transversal H' telle que $H' \subset H$.
H est appelé transversal minimum s'il n'existe pas de transversal H' telle que $|H'| < |H|$.

Exemple 4.3. *Soit $\Phi = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$ un ensemble de formules propositionnelles tel que $\mathcal{M}(\mathcal{F}_1) = \{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$, $\mathcal{M}(\mathcal{F}_2) = \{\mathcal{I}_1, \mathcal{I}_4\}$ et $\mathcal{M}(\mathcal{F}_3) = \{\mathcal{I}_3, \mathcal{I}_5\}$. $H = \{\mathcal{I}_1, \mathcal{I}_3\}$ est un transversal minimum et irréductible pour l'ensemble des modèles de Φ.*

Dans notre adaptation, nous choisissons le cluster le plus mauvais à diviser selon la mesure de qualité suivante. L'intuition derrière ce choix est d'augmenter le nombre de modèles partagés dans ce cluster choisi pour la division.

Définition 4.10. *Soit $\mathcal{C}_i = \{\mathcal{F}_{1_i}, \ldots, \mathcal{F}_{n_i}\}$ un cluster de n_i formules propositionnelles. Nous définissons la qualité de $\mathcal{C}_i$ comme suit :*

$$\mathcal{Q}(\mathcal{C}_i) = \frac{|\mathcal{M}(\mathcal{F}_{1_i} \wedge \cdots \wedge \mathcal{F}_{n_i})|}{|\mathcal{M}(\mathcal{F}_{1_i} \vee \cdots \vee \mathcal{F}_{n_i})|}$$

La qualité d'un cluster est obtenue en étendant la mesure de similarité entre deux formules à un ensemble de formules. En effet, un ensemble de formules est qualifié pour être de mauvaise qualité, lorsque

ses formules admettent un grand nombre de modèles tout en partageant un petit nombre de modèles. Par conséquent, le plus mauvais cluster est obtenu comme suit :

$$\mathcal{C}_i^* = \underset{\mathcal{C}_i \in \mathcal{C}}{\operatorname{argmin}}\ \mathcal{Q}(\mathcal{C}_i)$$

Définition 4.11. *Soit Φ un ensemble de formules propositionnelles et $\mathcal{I}$ un modèle. Nous définissons le sous-ensemble de formules de Φ partageant le modèle $\mathcal{I}$ comme $\mathcal{S}(\mathcal{I}, \Phi) = \{\mathcal{F} \in \Phi \mid \mathcal{I} \vDash \mathcal{F}\}$*

Algorithm 17: Algorithme hiérarchique descendant basé sur une représentation explicite des modèles pour le clustering des formules propositionnelles

Input: Un ensemble de formules $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ et un entier $k \geqslant 1$
Output: Une partition $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$

1 $H \leftarrow transversalMin(\{\mathcal{M}(\mathcal{F}_1), \ldots, \mathcal{M}(\mathcal{F}_n)\})$;
2 **if** *($|H| > k$)* **then**
3 **return** $\emptyset$;
4 **else**
5 $\mathcal{C} \leftarrow \{\Phi\}$;
6 **while** *($|\mathcal{C}| \,!= k$)* **do**
7 $\mathcal{C}_i^* = \{\mathcal{F}_{i_1} \ldots \mathcal{F}_{i_{n_i}}\} \leftarrow \underset{\mathcal{C}_i \in \mathcal{C}, |\mathcal{C}_i| > 1}{\arg\min}\ \mathcal{Q}(\mathcal{C}_i)$, $\triangleright\ n_i = |\mathcal{C}_i^*|$;
8 $M = \mathcal{M}(\mathcal{F}_{i_1}) \cap \cdots \cap \mathcal{M}(\mathcal{F}_{i_{n_i}})$;
9 $\Omega = \{\mathcal{M}(\mathcal{F}_{i_1}) \setminus M, \ldots, \mathcal{M}(\mathcal{F}_{i_{n_i}}) \setminus M\}$;
10 $H \leftarrow transversalMin(\Omega)$;
11 $\forall \mathcal{I} \in H, \Psi_{\mathcal{I}} \leftarrow \mathcal{S}(\mathcal{I}, \mathcal{C}_i^*)$;
12 **if** *($|\mathcal{C}| + |H| - 1 > k$)* **then**
13 $\Psi \leftarrow merge(\{\Psi_{\mathcal{I}_1}, \ldots, \Psi_{\mathcal{I}_{|\mathcal{C}|+|H|-1-k}}\})$;
14 $\mathcal{C} \leftarrow (\mathcal{C} \setminus \mathcal{C}_i^*) \cup \{\Psi\} \cup \{\Psi_{\mathcal{I}_{|\mathcal{C}|+|H|-k}}, \ldots, \Psi_{\mathcal{I}_{|H|}}\}$
15 **else**
16 $\mathcal{C} \leftarrow (\mathcal{C} \setminus \mathcal{C}_i^*) \cup \{\Psi_{\mathcal{I}_1}, \ldots, \Psi_{\mathcal{I}_{|H|}}\}$
17 $\mathcal{C} \leftarrow eliminateOverlap(\mathcal{C})$;
18 **return** $\mathcal{C}$

Décrivons maintenant comment les *transversaux* minimum peuvent être utilisés pour obtenir un clustering d'un ensemble Φ de formules propositionnelles. Pour construire des clusters cohérents, l'algorithme 17 commence par le calcul d'un *transversal* minimum H de l'ensemble des ensembles de modèles des formules dans Φ (ligne 1). L'idée principale derrière notre algorithme est d'utiliser les modèles du *transversal* minimum calculé pour diviser un cluster en plusieurs clusters cohérents. Chaque groupe est obtenu en sélectionnant pour chaque modèle $\mathcal{I}$ du *transversal* minimum, l'ensemble des formules admettant $\mathcal{I}$ comme modèle. De cette manière, les formules des clusters obtenus après la division partagent au moins un modèle.

Si la taille du *transversal* minimum H est supérieure à k, alors aucun clustering n'est possible, et l'algorithme renvoie un ensemble vide (ligne 3), sinon, un clustering cohérent peut être obtenu. Dans ce dernier cas, l'algorithme commence par un clustering $\mathcal{C}$ où toutes les formules de Φ sont regroupées dans un seul cluster (ligne 6). Ensuite, nous commençons un processus itératif de division (lignes 7-20), jusqu'à la génération de k clusters. À chaque itération, nous choisissons un cluster (ligne 8) qui a la plus

mauvaise qualité (voir Définition 4.10) pour l'étape de division. Ensuite, nous construisons Ω l'ensemble des ensembles de modèles des formules contenues dans le cluster sélectionné, tout en supprimant l'ensemble des modèles communs M (lignes 9-10). Un *transversal* minimum H de Ω est ensuite calculé (ligne 11).

Il est important de noter qu'en supprimant de chaque formule du cluster sélectionné pour la division, tous les modèles communs M, nous évitons de tomber sur un *transversal* minimum trivial de taille 1 parmi les modèles partagés.

Maintenant, nous utilisons l'ensemble H pour diviser le cluster choisi $\mathcal{C}_i^*$ en $|H|$ clusters (ligne 12). En effet, pour chaque modèle $\mathcal{I}$ dans H, on associe un cluster $\Psi_{\mathcal{I}}$ constitué de formules de $\mathcal{C}_i^*$ partageant le modèle $\mathcal{I}$. De cette façon, nous préservons la propriété de cohérence sur chaque nouveau cluster $\Psi_{\mathcal{I}}$.

Ensuite, nous substituons dans $\mathcal{C}$ le cluster de mauvaise qualité $\mathcal{C}_i^*$ par le nouvel ensemble de clusters (ligne 18). Cependant, cette étape n'est appliquée que lorsque la taille du nouveau cluster ne dépasse pas k (ligne 13) ; Sinon pour obtenir exactement k clusters, nous fusionnons (fonction `merge`) les premiers $|\mathcal{C}| + |H| - (k+1)$ de ces nouveaux clusters (ligne 14) avant d'appliquer la substitution (ligne 15).

Notons qu'après l'étape de division (ligne 12), une formule peut appartenir à plusieurs nouveaux clusters. La raison vient du fait qu'une formule donnée peut partager plusieurs modèles du *transversal* minimum. Par conséquent, à la fin de la boucle, l'algorithme fournit des clusters avec une possibilité de chevauchements. Une dernière étape est ensuite effectuée pour produire des clusters qui ne se chevauchent pas (ligne 20 - fonction `eliminOverlap`).

La dernière étape est effectuée en utilisant un algorithme glouton pour l'élimination des chevauchements. Pour chaque formule apparaissant dans plusieurs clusters, nous la conservons uniquement dans le cluster de la meilleure qualité, tout en l'enlevant des autres clusters. Évidemment, en fonction des applications, le clustering avec chevauchements peut être le plus approprié. Dans ce cas, il suffit de passer l'appel à la fonction d'élimination de chevauchement. Si les chevauchements sont autorisés, le problème de clustering est appelé clustering avec chevauchements.

L'algorithme 17 implique $\mathcal{O}(n)$ appels à un oracle d'énumération de modèles (ligne 1), et $\mathcal{O}(k)$ appels à un calcul d'un *transversal* minimum (lignes 1 et 10).

Donnons maintenant quelques propriétés intéressantes de notre algorithme basé sur la division pour le clustering des formules propositionnelles. La première propriété indique la correction de notre algorithme. En d'autres termes, l'algorithm 17 est capable de trouver un clustering valide pour tout problème de clustering cohérent.

Proposition 4.1 (Correction)**.** *Si $\mathcal{P}(k, \Phi)$ est cohérent, alors l'algorithme 17 produit un clustering.*

Démonstration. Notons d'abord que l'étape de division est basée sur le calcul d'un *transversal* minimum de l'ensemble des ensembles de modèles d'un cluster donné $\mathcal{C}$. Un tel cluster est divisé en plusieurs clusters obtenus en sélectionnant pour chaque modèle $\mathcal{I}$ du transversal, les formules admettant $\mathcal{I}$ comme modèle. Si l'opération de substitution conduit à un nombre de clusters supérieur à k, certains d'entre eux sont fusionnés (lignes 14). Le nouveau cluster obtenu par une telle opération de fusion est également cohérent (ligne 14). En effet, le cluster sélectionné est cohérent, ce qui signifie que toutes ses formules partagent au moins un modèle. Par conséquent, les formules impliquées dans l'opération de fusion admettent également un modèle commun. Pour compléter la preuve sur la cohérence du clustering

obtenu, considérons le cas où le clustering n'est pas possible. L'algorithme commence par calculer un *transversal* minimum H de l'ensemble des ensembles de modèles du cluster original contenant toute les formules (ligne 3). Si la taille de H est supérieure à k, aucun clustering n'est possible et nous retournons un ensemble vide. Cependant, dans le cas positif (lignes 6-18), en utilisant H nous garantissons comme expliqué ci-dessus l'obtention d'un ensemble de clusters cohérents à partir des modèles de H. La taille k du clustering est vérifiée par des tests effectués aux lignes 4 et 13. Évidemment, l'union des clusters contient toutes les formules originales. □

La deuxième propriété permet d'affirmer que deux formules seront dans les même clusters lorsque les chevauchements entre clusters sont autorisés, si l'une est une conséquence logique de l'autre.

Proposition 4.2. *Soit $\mathcal{P}(k, \Phi)$ un prblème de clustering avec chevauchement, $\mathcal{C}$ est un clustering de $\mathcal{P}(k, \Phi)$ et $\mathcal{F}_1 \in \Phi$ et $\mathcal{F}_2 \in \Phi$. Si $\mathcal{F}_1 \models \mathcal{F}_2$ alors $\forall \mathcal{C}_i \in \mathcal{C}$, si $\mathcal{F}_1 \in \mathcal{C}_i$ alors $\mathcal{F}_2 \in \mathcal{C}_i$.*

Démonstration. Nous avons $\mathcal{M}(\mathcal{F}_1) \subseteq \mathcal{M}(\mathcal{F}_2)$. A partir de notre définition de l'étape de division en utilisant un *transversal* minimum H, supposons que $\mathcal{F}_1$ appartient à un cluster partageant un modèle $\mathcal{I} \in H$. Comme $\mathcal{F}_2$ partage le même modèle $\mathcal{I}$, donc $\mathcal{F}_2$ appartient aussi à ce cluster. Cependant, l'inverse n'est pas vrai car $\mathcal{F}_2$ peut appartenir à un cluster partageant un modèle $\mathcal{I}$ qui n'est pas un modèle de $\mathcal{F}_1$. Par conséquent, $\mathcal{F}_1$ ne peut appartenir au même cluster dans ce cas. □

La dernière propriété nous permet d'affirmer que deux formules équivalentes seront dans les mêmes clusters lorsque les chevauchements entre clusters sont autorisés.

Proposition 4.3. *Soit $\mathcal{P}(k, \Phi)$ un prblème de clustering avec chevauchement, $\mathcal{C}$ est un clustering de $\mathcal{P}(k, \Phi)$ et $\mathcal{F}_1, \mathcal{F}_2 \in \Phi$. Si $\mathcal{F}_1 \equiv \mathcal{F}_2$ alors $\forall \mathcal{C}_i \in \mathcal{C}$, $\mathcal{F}_1 \in \mathcal{C}_i$ ssi $\mathcal{F}_2 \in \mathcal{C}_i$.*

Démonstration. Cette proposition est une conséquence directe de la proposition 4.2, car deux formules équivalentes sont des conséquences logiques l'un de l'autre. □

4.4 Encodage SAT pour un clustering cohérent de formules propositionnelles

Comme discuté dans les sections précédentes, lorsque les formules propositionnelles sont représentées par leurs modèles, notre algorithme qui se base sur la représentation par modèles requiert $\mathcal{O}(n)$ appels à un oracle d'énumération de modèles, pour calculer l'ensemble des modèles de chaque formule. Un tel ensemble de modèles peut être de taille exponentielle dans le pire des cas. En plus de ces limitations, il faut également calculer un *transversal* minimum d'un ensemble d'ensembles de modèles ($\mathcal{O}(k)$ appels).

Dans cette section, nous présentons une approche alternative qui ne nécessite qu'un nombre linéaire d'appels à un oracle SAT. Ceci réduit considérablement la complexité globale de notre algorithme. À cette fin, nous introduisons un encodage basé sur SAT qui permet de trouver un clustering cohérent borné d'un ensemble donné de formules propositionnelles. En d'autres termes, nous associons à chaque ensemble de formules propositionnelles Φ et un entier positif k une instance SAT qui est satisfiable si et seulement s'il existe un partitionnement de Φ en k clusters cohérents. Tout d'abord, une telle formulation SAT nous permet d'éviter les appels à un oracle d'énumération. De plus, un *transversal* minimum peut être calculé en utilisant SAT et nous allons le montrer par la suite.

Soit $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ un ensemble de formules propositionnelles et k un entier positif. Pour définir notre encodage, nous associons à chaque variable propositionnelle p apparaissant dans Φ un ensemble de

k variables propositionnelles, notées $p^1, \ldots, p^k$. Ensuite, pour toute formule $\mathcal{F}_i \in \Phi$ et $j \in \{1, \ldots, k\}$, nous utilisons $\mathcal{F}_i^j$ pour désigner la formule obtenue à partir de $\mathcal{F}_i$ en remplaçant chaque variable propositionnelle p par la nouvelle variable p^j. La formule $\mathcal{F}_i^j$ est utilisée pour modéliser le fait que $\mathcal{F}_i$ est dans le $j^{ème}$ cluster.

La formule suivante exprime que chaque formule dans Φ doit être vraie dans au moins un cluster cohérent :

$$\bigwedge_{i=1}^{n}(\bigvee_{j=1}^{k}\mathcal{F}_i^j) \tag{4.3}$$

On peut facilement voir que (4.3) est satisfiable si et seulement si Φ peut être partitionné en k clusters cohérents. Il est important de noter que dans un modèle de (4.3) une formule peut appartenir à plusieurs clusters. Pour obtenir un clustering cohérent borné à partir d'un modèle $\mathcal{I}$, il suffit de considérer pour chaque formule $\mathcal{F}_i \in \Phi$ un seul entier positif j dans l'ensemble $\{1 \leqslant j \leqslant k \mid \mathcal{I}(\mathcal{F}_i^j) = 1\}$. Ce problème peut être évité par reformulation. A cette fin, nous associons à chaque formule $\mathcal{F}_i$ dans Φ un ensemble de k nouvelles variables propositionnelles, notées $q^1_{\mathcal{F}_i}, \ldots, q^k_{\mathcal{F}_i}$. La variable $q^j_{\mathcal{F}_i}$ est utilisée pour représenter le fait que $\mathcal{F}_i$ est dans le j^{me} cluster en utilisant la formule suivante :

$$\bigwedge_{i=1}^{n}(\bigwedge_{j=1}^{k} q^j_{\mathcal{F}_i} \Leftrightarrow \mathcal{F}_i^j) \tag{4.4}$$

Ensuite, pour exprimer que chaque formule de Φ appartient à un seul cluster cohérent, nous utilisons la formule suivante :

$$\bigwedge_{i=1}^{n}(\sum_{j=1}^{k} q^j_{\mathcal{F}_i} = 1) \tag{4.5}$$

Notons que (4.5) implique la fameuse contraintes AtMostOne (AMO). En effet, $\sum_{j=1}^{k} q^j_{\mathcal{F}_i} = 1$ est équivalente à $(\bigvee_{1\leqslant j\leqslant k} q^j_{\mathcal{F}_i}) \wedge AMO(\{q^1_{\mathcal{F}_i}, \ldots, q^k_{\mathcal{F}_i}\})$. Il existe dans la littérature des encodages SAT linéaires de la contrainte AtMostOne, comme l'encodage séquentiel de Sinz [Sin05] qui nécessite $n-1$ variables auxiliaires et $3n-4$ clauses, où n est le nombre de variables (voir chapitre 1).

Notre deuxième encodage SAT du problème de clustering cohérent et borné $\mathcal{P}(k, \Phi)$ est défini par la formule $\mathcal{P}_{SAT}(k, \Phi) = (4.4) \wedge (4.5)$. A partir d'un modèle $\mathcal{I}$ de $\mathcal{P}_{SAT}(k, \Phi)$, un clustering peut être facilement extrait. En effet, si $m(q^j_{\mathcal{F}_i}) = 1$ alors $\mathcal{F}_i \in \mathcal{C}_j$ sinon $\mathcal{F}_i \notin \mathcal{C}_j$. Contrairement au premier encodage (4.3), un modèle de $\mathcal{P}_{SAT}(k, \Phi)$ associe à chaque formule un seul cluster.

Présentons maintenant une adaptation de notre approche de clustering basée sur la division décrite dans l'algorithme 17 pour traiter les formules propositionnelles sans avoir besoin de calculer leurs modèles.

Définition 4.12. *Soit $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ un ensemble de formules propositionnelles. $\mathcal{C}$ est un clustering cohérent minimum s'il n'existe pas de clustering cohérent $\mathcal{C}'$ tel que $|\mathcal{C}'| < |\mathcal{C}|$.*

Notons que dans l'algorithme 18, nous utilisons l'encodage SAT précédent pour calculer un clustering cohérent minimum $\mathcal{C}$ d'un ensemble de formules propositionnelles Φ (lignes 1 et 4 - appel à $minClusteringSAT$). La cardinalité de H est le nombre minimum de clusters (h) qui peuvent exister. Il peut être calculé comme suit : en commençant par $h = 1$, on appelle successivement un solveur SAT sur la formule $\mathcal{P}_{SAT}(h, \Phi)$, tout en augmentant h jusqu'à ce qu'on trouve un modèle. Soit $\mathcal{I}$ le modèle trouvé, l'ensemble de clusters $\mathcal{C}$ peut être facilement extrait de $\mathcal{I}$ en regardant les valeurs de vérité des variables $y^j_{\mathcal{F}_i}$ dans $\mathcal{I}$.
Notons que le nombre minimum de clusters ainsi calculé correspond à la taille du *transversal* minimum de l'ensemble des ensembles de modèles des formules propositionnelles.

Proposition 4.4. *Soit $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ un ensemble de formules propositionnelles et H un transversal minimum de l'ensemble $\{\mathcal{M}(\mathcal{F}_1), \ldots, \mathcal{M}(\mathcal{F}_n)\}$ alors $|H|$ est le nombre minimum de clusters cohérents de Φ.*

Démonstration. Soit $H = \{\mathcal{I}_1, \ldots, \mathcal{I}_{|H|}\}$ un *transversal* minimum de l'ensemble $\{\mathcal{M}(\mathcal{F}_1), \ldots, \mathcal{M}(\mathcal{F}_n)\}$. Ceci signifie qu'il n'y a pas de *transversal* H' telle que $|H'| < |H|$. À partir de H nous pouvons construire un clustering cohérent $\mathcal{C} = \mathcal{C}_1, \ldots, \mathcal{C}_{|H|}$ tel que $\forall i (1 \leqslant i \leqslant |H|), \mathcal{C}_i = \{\mathcal{F} \in \Phi, \mathcal{I}_i \in \mathcal{M}(\mathcal{F})\}$. Supposons que $|\mathcal{C}|$ n'est pas le nombre minimum de clusters cohérents. Cela signifie qu'il existe un clustering cohérent $\mathcal{C}'$ de Φ tel que $|\mathcal{C}'| < |\mathcal{C}| = |H|$. Soit $\mathcal{C}' = \mathcal{C}'_1, \ldots, \mathcal{C}'_{|\mathcal{C}'|}$ un tel clustering cohérent. Comme chaque $\mathcal{C}'_i$ est consistent alors nous pouvons construire un *transversal* $H' = \{\mathcal{I}'_1, \ldots, \mathcal{I}'_{|\mathcal{C}'|}\}$ où $\forall \mathcal{F} \in \mathcal{C}'_i, \mathcal{I}'_i \in \mathcal{M}(\mathcal{F})$. Comme $|\mathcal{C}'| < |H|$, alors H n'est pas un *transversal* minimum. Nous obtenons donc une contradiction. Par conséquent, $|\mathcal{C}| = |H|$ est le nombre minimum de clusters cohérents. □

Algorithm 18: Algorithme hiérarchique descendant basé sur SAT pour le clustering des formules propositionnelles

Input: Un ensemble de formules $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ et un entier $k \geqslant 1$
Output: Un ensemble de clusters $\mathcal{C} = \{C_1, \ldots, C_k\}$

1 $\mathcal{C} \leftarrow minClusteringSAT(\Phi)$;
2 **while** *($|\mathcal{C}| < k$)* **do**
3 $\quad \mathcal{C}^*_i \leftarrow \arg\min_{\mathcal{C}_i \in \mathcal{C}} \mathcal{Q}(\mathcal{C}_i)$;
4 $\quad H = \{\Psi_1, \ldots, \Psi_{|H|}\} \leftarrow minClusteringSAT(\mathcal{C}^*_i)$;
5 $\quad$ **if** *($|\mathcal{C}| + |H| - 1 > k$)* **then**
6 $\quad\quad \Psi \leftarrow merge(\{\Psi_1, \ldots, \Psi_{|\mathcal{C}|+|H|-1-k}\})$;
7 $\quad\quad \mathcal{C} \leftarrow (\mathcal{C} \setminus \mathcal{C}^*_i) \cup \{\Psi\} \cup \{\Psi_{|\mathcal{C}|+|H|-k}, \ldots, \Psi_{|H|}\}$
8 $\quad$ **else**
9 $\quad\quad \mathcal{C} \leftarrow (\mathcal{C} \setminus \mathcal{C}^*_i) \cup H$
10 $\mathcal{C} \leftarrow eliminateOverlap(\mathcal{C})$;
11 **return** $\mathcal{C}$

Comme nous pouvons le constater, l'algorithme 18 est obtenu à partir de l'algorithme 17 en remplaçant le calcul du *transversal* minimum par le calcul d'un clustering cohérent minimum. Grâce à la proposition 4.4, et de la même manière que l'algorithme 17, les propositions 4.1, 4.3 et 4.2 sont vérifiées.

Encoder le problème de clustering cohérent et borné en utilisant SAT nous permet d'éviter $\mathcal{O}(n)$ appels à un oracle d'énumération de modèles, tout en utilisant des solveurs SAT efficaces au lieu d'utiliser un algorithme de calcul d'un *transversal* minimum.

4.5 Expérimentations

Dans cette section, nous avons effectué une évaluation expérimentale de la performance de nos algorithmes hiérarchiques descendant « divisive » et ascendant « agglomerative » pour le clustering d'un ensemble de formules propositionnelles. Notre objectif est d'évaluer la faisabilité et l'efficacité de l'approche proposée.

Nous avons effectué nos expériences sur une machine avec processeur Intel Core2 Quad de 2,66 GHz et 8G de RAM. Notre premier objectif est de comparer les performances de nos algorithmes hiérarchiques descendant et ascendant. À cette fin, nous avons considéré les jeux de données suivants :

1. *Bases de données transactionnelles* : Nous considérons trois jeux de données suivants : `splice`, `australian-credit`, and `german-credit` [2]. Nous considérons chaque jeu de données comme un ensemble de transactions, où chaque transaction est une formule (un ensemble de modèles). Par conséquent, un item est considéré comme un modèle.
2. *Préférences de voitures* : Un ensemble de 60 formules (ensembles de modèles préférés) encodant les préférences de voiture des utilisateurs sont considérés [ASBP13] [3]. Les voitures sont décrites par les attributs suivants et leurs valeurs correspondantes :
   — *Type de carrosserie* : Sedan, SUV
   — *Capacité moteur* : 2.5L, 3.5L, 4.5L, etc.
   — *Transmission* : Manuelle, Automatique
   — *Carburant consommé* : Hybride, Non-Hybride
   Nous avons considéré ici vingt modèles distincts. Chaque modèle est décrit par un ensemble de quatre valeurs d'attributs. Un questionnaire de préférences utilisateur contenant 38 paires de modèles de la forme (m, m') générées aléatoirement est présenté à 60 utilisateurs. Les préférences des utilisateurs de la forme $(m_i \succ m_j)$ sont ensuite collectées. Par conséquent, pour chaque utilisateur, nous obtenons un ensemble de 38 préférences utilisateur binaires induisant des relations binaires sur des modèles. Pour définir une formule propositionnelle à partir de chaque relation de préférence d'un utilisateur, nous encodons l'ensemble de ses modèles préférés. Ces modèles sont obtenus en représentant chaque relation de préférence binaire sous la forme d'un graphe où chaque nœud correspond à un modèle et les arêtes encodent cette relation. L'ensemble des modèles préférés est ensuite obtenu en collectant les modèles correspondant aux nœuds sans arêtes entrantes (nœuds sources).

Les figures 4.4, 4.5 et 4.6 montrent les performances des méthodes de clustering hiérarchiques ascendante (Algorithme 16) et descendante (Algorithme 17) sur le problème de clustering des données transactionnelles. Premièrement, notre algorithme descendant surpasse l'algorithme hiérarchique ascendant sur `splice` et `germen-credit`. Sur les données de `australian-credit`, l'approche ascendante est légèrement meilleure. Néanmoins, comme illustré dans la section 4.2.2, l'algorithme hiérarchique ascendant est incapable de trouver un clustering tout le temps. Ceci est le cas sur le jeu de données `splice`, où une telle approche ne peut pas fournir de solution de clustering lorsque le nombre de clusters désiré est inférieur à 84.

La figure 4.7 montre les performances des algorithmes hiérarchiques descendant et ascendant sur le clustering des préférences automobiles. Nous illustrons l'évolution de la qualité de clustering (AEC) (voir définition 4.8) des deux algorithmes en fonction du nombre de clusters k, varié de 6 à 20. Notons

2. https ://dtai.cs.kuleuven.be/CP4IM/
3. http://users.cecs.anu.edu.au/~u4940058/CarPreferences.html

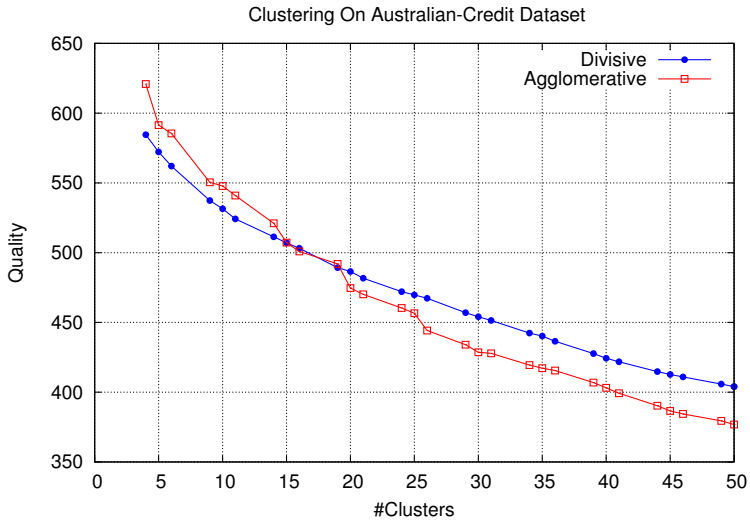


FIGURE 4.4 – Algorithme hiérarchiques : ascendant vs descendant sur `australian-credit`

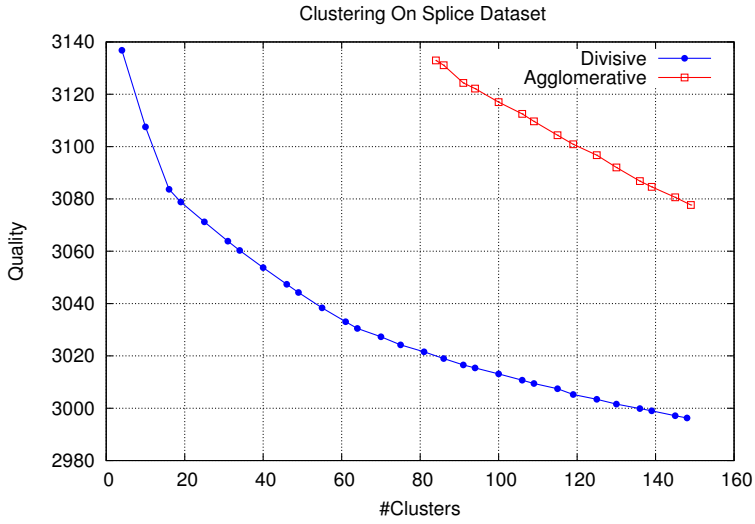


FIGURE 4.5 – Algorithme hiérarchiques : ascendant vs descendant sur `splice`

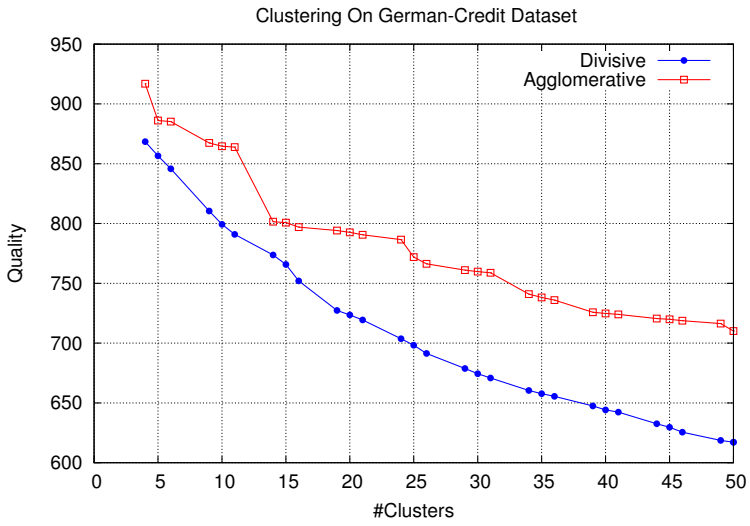


FIGURE 4.6 – Algorithme hiérarchiques : ascendant vs descendant sur `german-credit`

que pour le jeu de données considéré, les deux algorithmes sont capables de trouver une solution en moins d'une seconde et l'approche descendante est légèrement meilleure. Une autre remarque est que 7 représente le nombre minimum de clusters cohérents possibles en utilisant l'algorithme divisif. Dans ce cas, l'approche ascendante n'est pas capable de trouver un clustering de taille 7 contrairement à celle descendante.

Pour étudier la capacité de passage à l'échelle de notre approche, nous élargissons nos expériences du problème précédent en étudiant le clustering d'un ensemble de formules issues d'un sondage aléatoire de 100 à 1000 participants où chaque participant est invité à donner ses préférences. Les questions du sondage sont organisées en quatre niveaux. Au premier niveau, le participant est invité à sélectionner ses 3 options préférées parmi 5. Selon les préférences du participant, il est invité à sélectionner d'autres préférences du deuxième niveau et ainsi de suite jusqu'au dernier niveau (niveau 4). Par exemple, supposons que dans le premier niveau, nous considérons un ensemble S de cours (par exemple, l'intelligence artificielle, la fouille de données, les bases de données, les réseaux et la programmation Web). Un étudiant sélectionne trois cours de S (niveau 1). Ensuite, pour chaque cours sélectionné, il choisit des chapitres (niveau 2), et ainsi de suite. Les préférences de chaque participant sont encodées sous la forme d'une formule propositionnelle (les formules résultantes ont entre 567 et 1813 modèles). L'algorithme 18 est ensuite exécuté pour obtenir des clusters. L'approche hiérarchique ascendante n'est pas considérée puisqu'elle ne peut pas garantir de trouver une solution de clustering si elle existe.

La figure 4.8 décrit la variation de la qualité pour 1000 participants lorsque le nombre de clusters est varié. L'erreur est inversement proportionnelle à k. Ceci est intuitif puisque les préférences sont générées aléatoirement. De plus, le temps nécessaire pour obtenir un clustering, figure 4.10, ne dépasse pas 100 secondes pour toutes les valeurs de k. Cela montre que notre approche passe bien à l'échelle. Enfin, nous étudions l'évolution du temps nécessaire pour trouver un clustering lorsque le nombre de clusters est fixé à 20 et que le nombre de participants varie de 100 à 1000 (Figure 4.9). Là encore, le temps nécessaire est raisonnable, c'est-à-dire inférieur à 100 secondes.

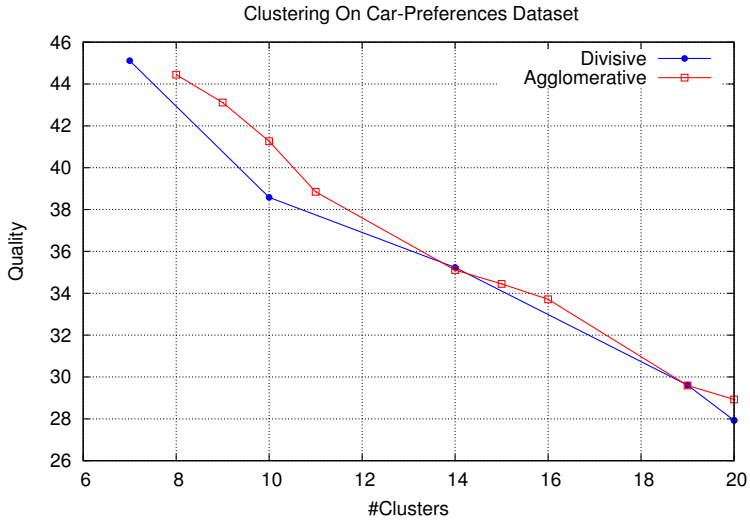


FIGURE 4.7 – Qualité minimum : préférences automobiles

4.6 Conclusion

Dans ce chapitre, nous avons introduit le concept de clustering cohérent de formules propositionnelles. Nous avons montré comment les algorithmes k-$means$ et l'algorithme hiérarchique ascendant peuvent être adaptés à ce nouveau cadre. Ensuite, nous avons proposé deux nouvelles solutions. La première suppose que l'ensemble de modèles de chaque formule est donné et utilise la notion de transversal dans un algorithme hiérarchique descendant pour générer efficacement un clustering cohérent. Dans la deuxième solution, nous avons proposé un encodage SAT pour l'algorithme descendant qui fait un nombre linéaire d'appels à un oracle #SAT pour compter les ensembles de modèles pendant les étapes de clustering. Enfin, des expérimentations préliminaires ont été effectuées pour montrer la faisabilité de notre approche.

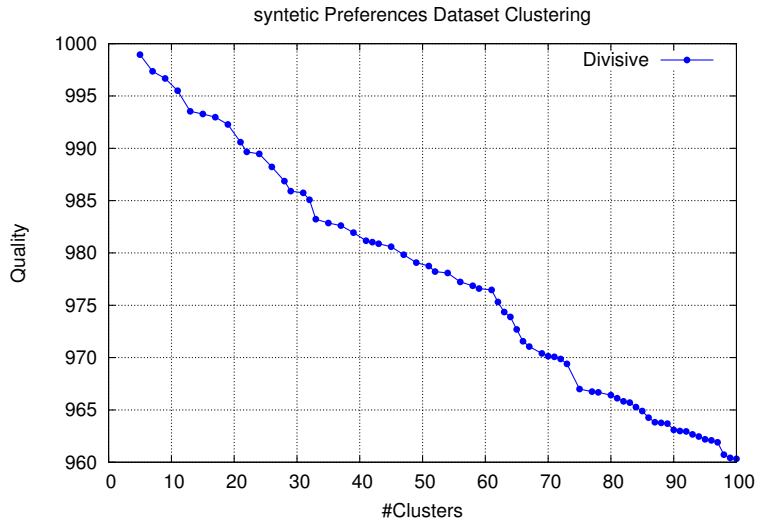


FIGURE 4.8 – Qualité minimum :

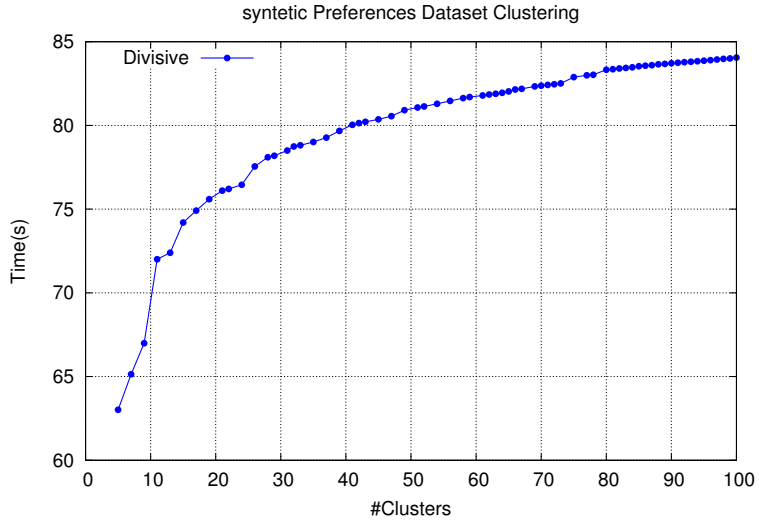


FIGURE 4.9 – Temps vs #Participants

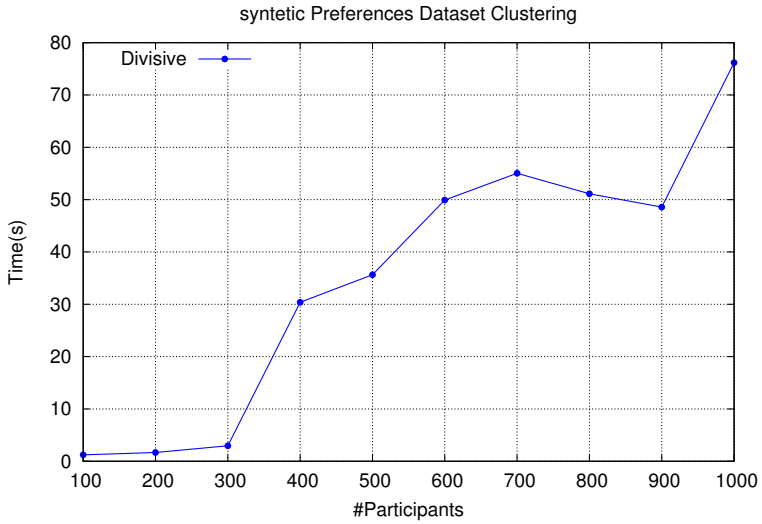


FIGURE 4.10 – Temps vs #Clusters

Chapitre 5

Satisfiabilité pour la fouille de règles d'association

Sommaire

L'extraction de règles d'association a connu de nombreux développements théoriques et algorithmiques. Il y a eu de nombreuses contributions pour développer de bonnes théories [HHR10] ainsi que des algorithmes rapides pour l'extraction des règles d'association. Parmi ces algorithmes, Apriori [AS94] et FP-Growth [HPYM04] sont les plus connus. Tous ces algorithmes qui calculent les règles d'association partagent une même méthodologie en deux étapes. La première est consacrée à la recherche de tous les itemsets fréquents, et la seconde consiste à générer les règles avec une grande confiance en combinant ces itemsets fréquents.

Comme souligné dans [RNOH11], les contraintes font souvent partie de la spécification de plusieurs problèmes de fouille de données. Cette observation a conduit à un nouveau domaine de recherche actif et multidisciplinaire initié dans [RGN08], et permettant une fertilisation croisée entre la fouille de données et l'intelligence artificielle. Cette nouvelle approche offre un modèle de représentation flexible et déclaratif. En effet, dans la fouille de données, de nouvelles contraintes nécessitent souvent de nouvelles implémentations, alors que ces mêmes contraintes peuvent être facilement intégrées dans de tels modèles déclaratifs. Suivant cette tendance de recherche, nous proposons dans ce chapitre une nouvelle approche basée sur la satisfiabilité propositionnelle pour fouiller les règles d'association en une seule étape. La tâche est modélisée comme une formule propositionnelle dont les modèles correspondent aux règles à

Tid	Itemset
1	c d e f g
2	c d e f g
3	a b c d
4	a b c d f
5	a b c d
6	c e

TABLE 5.1 – Une base de données transactionnelles $\mathcal{D}$.

fouiller.

Comme le nombre de règles d'association peut augmenter rapidement, surtout quand on baisse les exigences de la fréquence, limiter le nombre de règles produites sans perte d'informations est une question très importante. Il a également été noté que certains des motifs rares, tels que les associations indirectes, fournissent des informations utiles sur les données. Dans notre deuxième contribution, nous considérons plusieurs variantes bien connues, destinées à surmonter ces deux limitations principales, à savoir la fouille de règles d'association fermées [TPBL00], la fouille de règles d'association indirectes [TKS00] et la fouille des règles d'association non redondantes [BPT+00, Zak04]. Notre objectif est de montrer la flexibilité et la déclarativités de notre approche.

5.1 Encodage SAT pour la fouille de règles d'association

Dans cette section, on s'intéresse à l'encodage du problème de fouille de règles d'association en SAT. Rappelons qu'étant donnés une base de données transactionnelles $\mathcal{D}$, un seuil minimum de support α et un seuil minimum de confiance β, une *règle d'association* dans $\mathcal{D}$ est un motif de la forme $X \rightarrow Y$ où X et Y sont deux itemsets qui vérifient les propriétés suivantes :

1. $X \neq \emptyset$ et $Y \neq \emptyset$.
2. X, Y sont deux itemsets disjoints : $X \cap Y = \emptyset$.
3. *Support* : Les deux itemsets X, Y apparaissent ensemble dans au moins α transaction dans $\mathcal{D}$.
4. *Confiance* : l'itemset Y apparaît dans au moins β transactions où X apparaît.

Le problème de fouille de règles d'association consiste plus précisément à calculer l'ensemble suivant : $\mathcal{MAR}(\mathcal{D}, \alpha, \beta) = \{X \rightarrow Y \mid X, Y \subseteq \Omega \wedge \mathcal{S}(X \rightarrow Y, \mathcal{D}) \geqslant \alpha \wedge \mathcal{C}onf(X \rightarrow Y, \mathcal{D}) \geqslant \beta\}$

Nous décrivons maintenant notre encodage fondé sur SAT pour le problème de fouille de règles d'association. L'idée de base consiste à utiliser des variables propositionnelles pour représenter la couverture des itemsets X et $X \cup Y$ pour chaque règle candidate $X \rightarrow Y$. Ces variables sont utilisées dans des inéquations linéaires 0/1 pour déterminer si le support et la confiance de la règle candidate sont supérieurs aux seuils minimaux spécifiés pour le support et la confiance. Soit $\mathcal{D} = \{(1, I_1), \ldots, (m, I_m)\}$ une base de données transactionnelles construite sur un ensemble d'items Ω, α un seuil minimum pour le support et β un seuil minimum pour la confiance. Pour représenter les deux itemsets de chaque règle candidate $X \rightarrow Y$, on associe deux variables propositionnelles x_a et y_a à chaque item a. Les variables de la forme x_a (resp. y_a) sont utilisées pour représenter l'antécédent (resp. conséquence) de chaque règle candidate. Ensuite, pour représenter la couverture de X et $X \cup Y$, on associe à chaque identifiant d'une transaction

$i \in \{1 \dots m\}$ deux variables propositionnelles p_i et q_i. Les variables de la forme p_i (resp. q_i) sont utilisées pour représenter la couverture de X (resp. $X \cup Y$). Plus précisément, étant donné une interprétation booléenne $\mathcal{I}$, la règle candidate est $X = \{a \in \Omega \mid \mathcal{I}(x_a) = 1\} \rightarrow Y = \{b \in \Omega \mid \mathcal{I}(y_b) = 1\}$, la couverture de X est $\{i \in \mathbb{N} \mid \mathcal{I}(p_i) = 1\}$, et la couverture de $X \cup Y$ est $\{i \in \mathbb{N} \mid \mathcal{I}(q_i) = 1\}$.

Nous présentons maintenant notre encodage basé sur SAT en utilisant les variables propositionnelles décrites précédemment.

Les deux clauses de la première formule expriment que X et Y ne sont pas vides.

$$(\bigvee_{a \in \Omega} x_a) \wedge (\bigvee_{a \in \Omega} y_a) \tag{5.1}$$

Exemple 5.1. *Soit $\mathcal{D}$ la base de données transactionnelles représentée par la Table 5.1, la formule 5.1 est donnée par les deux clauses suivantes :*

$$(x_a \vee x_b \vee x_c \vee x_d \vee x_e \vee x_f \vee x_g) \wedge (y_a \vee y_b \vee y_c \vee y_d \vee y_e \vee y_f \vee y_g)$$

La deuxième formule propositionnelle permet d'exprimer la contrainte $X \cap Y = \emptyset$, en imposant que x_a et y_a ne soient pas vraies en même temps.

$$\bigwedge_{a \in \Omega} (\neg x_a \vee \neg y_a) \tag{5.2}$$

Exemple 5.2. *Soit $\mathcal{D}$ la base de données transactionnelles représentée par la Table 5.1, la formule 5.2 correspond aux clauses binaires suivantes :*

$$\begin{array}{lll} (\neg x_a \vee \neg y_a) & (\neg x_d \vee \neg y_d) & \\ (\neg x_b \vee \neg y_b) & (\neg x_e \vee \neg y_e) & (\neg x_g \vee \neg y_g) \\ (\neg x_c \vee \neg y_c) & (\neg x_f \vee \neg y_f) & \end{array}$$

Pour obtenir la couverture de l'itemset X, on utilise la formule propositionnelle 5.3.

$$\bigwedge_{i=1}^{m} (\neg p_i \leftrightarrow \bigvee_{a \in \Omega \setminus I_i} x_a) \tag{5.3}$$

Dans cette formule, p_i est $faux$ ssi X contient un item qui n'appartient pas à la transaction i. Comme conséquence la couverture de X est $\{i \in \mathbb{N} \mid \mathcal{I}(p_i) = 1\}$.
Notons que la formule 5.3 est réécrite sous forme CNF comme suit :

$$\bigwedge_{i=1}^{m} (\ (p_i \vee \bigvee_{a \in \Omega \setminus I_i} x_a) \ \bigwedge_{a \in \Omega \setminus I_i} (\neg p_i \vee \neg x_a) \ )$$

Proposition 5.1. *Soit $\mathcal{D}$ une base de données transactionnelle et $\mathcal{I}$ une interprétation Booléenne de la formule (5.3). Alors, $\mathcal{I}$ est un modèle de (5.3) ssi $\{i \in \{1, \dots, m\} | \mathcal{I}(p_i) = 1\}$ est la couverture dans $\mathcal{D}$ de l'itemset $\{a \in \Omega | \mathcal{I}(x_a) = 1\}$.*

Exemple 5.3. *Soit $\mathcal{D}$ la base de données transactionnelles représentée par la Table 5.1, la formule 5.3 correspond aux clauses suivantes pour capturer la couverture de X :*

$(p_1 \vee x_a \vee x_b \vee x_d)$
$(\neg p_1 \vee \neg x_a)$
$(\neg p_1 \vee \neg x_b)$
$(\neg p_1 \vee \neg x_d)$
$(p_2 \vee x_a \vee x_b$
$(\neg p_2 \vee \neg x_a)$
$(\neg p_2 \vee \neg x_b)$
$(p_3 \vee x_e \vee x_f \vee x_g)$
$(\neg p_3 \vee \neg x_e)$
$(\neg p_3 \vee \neg x_f)$
$(\neg p_3 \vee \neg x_g)$
$(p_4 \vee x_e \vee x_g)$
$(\neg p_4 \vee \neg x_e)$
$(\neg p_4 \vee \neg x_g)$
$(p_5 \vee x_e \vee x_f \vee x_g)$
$(\neg p_5 \vee \neg x_e)$
$(\neg p_5 \vee \neg x_f)$
$(\neg p_5 \vee \neg x_g)$
$(p_6 \vee x_a \vee x_b \vee x_d \vee x_f \vee x_g)$
$(\neg p_6 \vee \neg x_a)$
$(\neg p_6 \vee \neg x_b)$
$(\neg p_6 \vee \neg x_d)$
$(\neg p_6 \vee \neg x_f)$
$(\neg p_6 \vee \neg x_g)$

De la même façon que la formule précédente, on utilise la formule suivante pour capturer la couverture de $X \cup Y$:

$$\bigwedge_{i=1}^{m}(\neg q_i \leftrightarrow \neg p_i \vee (\bigvee_{a \in \Omega \setminus I_i} y_a)) \tag{5.4}$$

Il est intéressant de noter que nous utilisons les variables propositionnelles p_i pour empêcher la réutilisation des variables de la forme x_a. Cela nous permet d'obtenir une formule plus compacte. La formule 5.4 est équivalente à la formule CNF suivante :

$$\bigwedge_{i=1}^{m}((q_i \vee \neg p_i \bigvee_{a \in \Omega \setminus I_i} y_a) \wedge (\neg q_i \vee p_i) \bigwedge_{a \in \Omega \setminus I_i} (\neg q_i \vee \neg y_a))$$

Proposition 5.2. *Soit $\mathcal{D}$ une base de données transactionnelle $\mathcal{I}$ une interprétation Booléenne de la formule (5.3)∧(5.4). Alors, $\mathcal{I}$ est un modèle de (5.3)∧(5.4) ssi $\{i \in \{1,\ldots,m\} | \mathcal{I}(p_i) = 1\}$ est la couverture dans $\mathcal{D}$ de l'itemset $\{a \in \Omega | \mathcal{I}(x_a) = 1\}$ et $\{i \in \{1,\ldots,m\} | \mathcal{I}(q_i) = 1\}$ est la couverture dans $\mathcal{D}$ de l'itemset $\{a \in \Omega | \mathcal{I}(x_a) = 1 \vee \mathcal{I}(y_a) = 1\}$.*

Exemple 5.4. *Soit $\mathcal{D}$ la base de données transactionnelles représentée par la Table 5.1, la formule 5.3 correspond aux clauses suivantes pour capturer la couverture de l'itemset $X \cup Y$:*

$(q_1 \vee \neg p_1 \vee y_a \vee y_b \vee y_d)$
$(\neg q_1 \vee p_1)$
$(\neg q_1 \vee \neg y_a)$
$(\neg q_1 \vee \neg y_b)$
$(\neg q_1 \vee \neg y_d)$
$(q_2 \vee \neg p_2 \vee y_a \vee y_b$
$(\neg q_2 \vee p_2)$
$(\neg q_2 \vee \neg y_a)$
$(\neg q_2 \vee \neg y_b)$
$(q_3 \vee \neg p_3 \vee y_e \vee y_f \vee y_g)$
$(\neg q_3 \vee p_3)$
$(\neg q_3 \vee \neg y_e)$
$(\neg q_3 \vee \neg y_f)$
$(\neg q_3 \vee \neg y_g)$
$(q_4 \vee \neg p_4 \vee y_e \vee y_g)$
$(\neg q_4 \vee p_4)$
$(\neg q_4 \vee \neg y_e)$
$(\neg q_4 \vee \neg y_g)$
$(q_5 \vee \neg p_5 \vee y_e \vee y_f \vee y_g)$
$(\neg q_5 \vee p_5)$
$(\neg q_5 \vee \neg y_e)$
$(\neg q_5 \vee \neg y_f)$
$(\neg q_5 \vee \neg y_g)$
$(q_6 \vee \neg p_6 \vee y_a \vee y_b \vee y_d \vee y_f \vee y_g)$
$(\neg q_6 \vee p_6)$
$(\neg q_6 \vee \neg y_a)$
$(\neg q_6 \vee \neg y_b)$
$(\neg q_6 \vee \neg y_d)$
$(\neg q_6 \vee \neg y_f)$
$(\neg q_6 \vee \neg y_g)$

La formule 5.5 exprime que le support de la règle candidate doit être supérieur ou égal au seuil spécifié α.

$$\sum_{i=1}^{m} q_i \geqslant \alpha \tag{5.5}$$

Cette formule signifie que le nombre de variables q_i ($i \in \{1 \ldots m\}$) assigné à 1 doit être supérieur ou égal à α, ce qui est équivalent au fait que le support de $X \cup Y$ est supérieur ou égal à $m \times \alpha$.

Enfin, nous décrivons la formule exprimant le fait que β est un seuil de confiance minimum :

$$\frac{\sum_{i=1}^{m} q_i}{\sum_{i=1}^{m} p_i} \geqslant \beta \tag{5.6}$$

On considère ici que β est donné en pourcentage $\beta\%$ tel que β est un entier positif. Ainsi, la formule précédente peut être réécrite comme suit :

$$100 * \sum_{i=1}^{m} q_i - \beta * \sum_{i=1}^{m} p_i \geqslant 0 \tag{5.7}$$

Les deux formules propositionnelles (5.5), (5.7) correspondent aux inégalités linéaires 0/1, généralement appelées contrainte de cardinalité et contrainte Pseudo Booléenne. Plusieurs auteurs se sont penchés sur la question de trouver un encodage CNF efficace des contraintes de cardinalité (e.g.[ANORC11, JSS14, War98]) ou Pseudo Booléenne (e.g. [BBR09, ES06, War98]). Pour plus de détails voir le chapitre 1.

Exemple 5.5. *Soit $\mathcal{D}$ la base de données transactionnelles représentée par la Table 5.1, les deux formules 5.5 et 5.7 correspondent au deux contraintes pseudo-Booléennes suivantes :*

$$q_1 + q_2 + q_3 + q_4 + q_5 + q_6 \geqslant \alpha$$

$$100 * \sum_{i=1}^{6} q_i - \beta * \sum_{i=1}^{6} p_i \geqslant 0$$

Nous utilisons $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ pour noter la conjonction des formules (5.1), (5.2), (5.3), (5.4), (5.5) et (5.7).

Proposition 5.3. *Soient $\mathcal{D}$ une base de données transactionnelles, α un seuil minimum de support, β un seuil minimum de confiance et $\mathcal{I}$ une interprétation Booléenne de $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$. Alors, $\mathcal{I}$ est un modèle de $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ ssi $\{a \in \Omega | \mathcal{I}(x_a) = 1\} \rightarrow \{a \in \Omega | \mathcal{I}(y_a) = 1\}$ est une règle d'association valide par rapport à α et β*

Démonstration. Partie $\Rightarrow$. En utilisant la proposition 5.2 nous savons que $\{i \in \{1, \ldots, m\} | \mathcal{I}(q_i) = 1\}$ est la couverture de $\{a \in \Omega | \mathcal{I}(x_a) = 1 \vee \mathcal{I}(y_a) = 1\}$ représentant la règle $r : X \rightarrow Y$. En effet, en utilisant la contrainte (5.4) nous savons que le support de r est supérieur ou égal à α. De plus, en utilisant la proposition 5.1 nous savons que $\{i \in \{1, \ldots, m\} | \mathcal{I}(p_i) = 1$ est la couverture de l'itemset $X = \{a \in \Omega | \mathcal{I}(x_a) = 1\}$ et en utilisant la contrainte (5.7) nous savons que la confiance de r est supérieure ou égale à β. En conséquence, r est une règle d'association.
Partie $\Leftarrow$. Soit $r : X \rightarrow Y$ une règle d'association dans $\mathcal{D}$. l'interprétation Booléenne $\mathcal{I}$ définie comme suit : $\forall a \in \Omega, \mathcal{I}(x_a) = 1$ ssi $a \in X$ et $\mathcal{I}(y_a) = 1$ ssi $a \in Y$ avec $\mathcal{I}(p_i) = 1$ ssi $i \in C(X, \mathcal{D})$ et $\mathcal{I}(q_i) = 1$ ssi $i \in C(X \cup Y, \mathcal{D})$. En utilisant les deux propositions 5.1 et 5.2, nous savons que $\mathcal{I}$ est un modèle de la formule (5.3)$\wedge$(5.4). De plus $\mathcal{I}$ est un modèle de la formule (5.5) car le support de r qui est égal à la somme des variables q_i est supérieur ou égal à α. De même pour la confiance. En conséquence, $\mathcal{I}$ est un modèle de $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$. □

Notons que le nombre de modèles de $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ est égal au nombre de règles d'associations. Cela vient tout particulièrement du faite que tout itemset n'a qu'une unique couverture, donc impossible de trouver une même règle deux fois.

5.2 Variantes de règles d'association

Dans cette section, notre objectif pour la modélisation et la résolution est de montrer les aspects déclaratifs et flexibles de l'approche basée sur SAT pour la fouille de règles d'association. À cette fin, nous considérons plusieurs variantes de règles d'association bien connues, à savoir les règles indirectes [TKS00], les règles fermées [TPBL00], les règles minimales non redondantes [BPT+00, Zak04]. La première vise à trouver des relations indirectes entre les items, tandis que les autres ont été proposées pour éviter les règles redondantes en utilisant une représentation condensée.

5.2.1 Règles d'association indirectes

Les associations indirectes sont largement utilisées pour construire des systèmes de recommandation web [Kaz09b]. Elles font référence à des paires d'éléments qui se produisent rarement ensemble, mais qui dépendent fortement de la présence d'un itemset médiateur.

Définition 5.1. *Soit $\mathcal{D}$ une base de données transactionnelles. Deux items a_0 et b_0 sont indirectement liés par l'intermédiaire d'un itemset M, appelé médiateur, en respectant un seuil de support maximum λ, un seuil de support minimum α et un seuil de dépendance au médiateur minimum β ssi les conditions suivantes sont satisfaites :*

- $\mathcal{S}(\{a_0\} \rightarrow \{b_0\}, \mathcal{D}) \leqslant \lambda$ *(condition de support sur la paire d'items)* [4].
- $\mathcal{S}(\{a_0\} \rightarrow M, \mathcal{D}) \geqslant \alpha$ *et* $\mathcal{S}(\{b_0\} \rightarrow M, \mathcal{D}) \geqslant \alpha$ *(Condition de Support sur M)*
- $\mathcal{D}ep(\{a_0\}, M, \mathcal{D}) \geqslant \beta$ *et* $\mathcal{D}ep(\{b_0\}, M, \mathcal{D}) \geqslant \beta$ *où* $\mathcal{D}ep(P, Q, \mathcal{D})$ *est une mesure de dépendance entre P et Q par rapport à $\mathcal{D}$ (Condition de dépendance)*

En d'autres termes, dans la fouille des règles indirectes, nous cherchons les paires d'items qui sont infréquentes (rares) mais qui sont impliquées séparément dans des règles d'association intéressantes avec la même conséquence.
Plusieurs mesures de dépendance entre deux itemsets X, Y ont été proposées, y compris la mesure IS [TKS02] et la confiance classique. La pertinence de ces mesures dépend de l'application cible. Dans ce travail, on a simplement utilisé la confiance de $X \rightarrow Y$ comme mesure de dépendance. Cette dernière a été utilisée dans [Kaz05] dans une application de recommandation web. En utilisant la confiance comme mesure de dépendance et un seuil minimum de confiance au lieu du seuil de dépendance médiateur, les deux conditions (support du médiateur et dépendance) dans la définition 5.1 peuvent être réécrites comme suit : $\{a_0\} \rightarrow M$ et $\{b_0\} \rightarrow M$ sont deux règles d'association valides en respectant les seuils α et β.

Exemple 5.6. *En considérant la base de données transactionnelles représentée par la table 5.1, les deux itemsets a et e sont indirectement liés par l'intermédiaire d pour un minimum de support égal à 2 et pour un support maximum de 1 et une confiance minimum de 1, car le support de l'itemset ae est égal à 0 et les deux règles $a \rightarrow d$ et $e \rightarrow d$ ont des support supérieurs à 2 et des confiances égales à 1.*

Afin de définir un encodage SAT pour le problème de fouille de règles d'association indirectes, nous devons utiliser des variables propositionnelles qui nous permettent de capturer la couverture de la règle d'association $\{a_0\} \rightarrow \{b_0\}$ et nous adaptons l'encodage $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ pour contraindre les antécédents des deux règles d'association $\{a_0\} \rightarrow M$ et $\{b_0\} \rightarrow M$ afin qu'ils contiennent qu'un seul item.

On utilise pour chaque item les variables propositionnelles $x_c^{a_0}$ et $x_c^{b_0}$ pour représenter a_0 et b_0 respectivement. On utilise le même ensemble de variables y_a pour chaque item a comme dans $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ afin de capturer les éléments du médiateur. De plus, on introduit des variables de la forme $p_i^{a_0}$ (resp. $p_i^{b_0}$)

4. on peut aussi écrire $\frac{\mathcal{S}(\{a_0, b_0\}, \mathcal{D})}{|\mathcal{D}|} \leqslant \lambda$

pour exprimer la couverture de l'item a_0 (resp. b_0) de la même façon que dans $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ et des variables de la forme $q_i^{a_0}$ et $q_i^{b_0}$ pour exprimer la couverture de $M \cup \{a_0\}$ et $M \cup \{b_0\}$ respectivement. Enfin, on introduit des variables de la forme r_i pour capturer la couverture de $\{a_0, b_0\}$.

Les deux formules suivantes permettent de capturer les règles $\{a_0\} \rightarrow M$ et $\{b_0\} \rightarrow M$:

$$\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta) \wedge (\sum_{a \in \Omega} x_a^{a_0} = 1) \ , \ \mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta) \wedge (\sum_{a \in \Omega} x_a^{b_0} = 1) \tag{5.8}$$

Où les variables de la forme x_a, p_i et q_i sont remplacées dans $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ par $x_a^{a_0}$, $p_i^{a_0}$et $q_i^{a_0}$ respectivement. La contrainte de cardinalité $\sum_{a \in \Omega} x_a^{a_0} = 1$ est utilisée pour exiger que l'antécédent des règles doit contenir un seul item. De la même façon que la formule (5.8), on utilise la formule suivante pour capturer la règle d'association $\{b_0\} \rightarrow M$:

$$\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta) \wedge (\sum_{a \in \Omega} x_a^{b_0} = 1) \tag{5.9}$$

Où les variables de la forme x_a, p_i et q_i sont remplacées dans $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ par $x_a^{b_0}$, $p_i^{b_0}$ et $q_i^{b_0}$ respectivement. Nous décrivons maintenant la formule qui permet de capturer la couverture de $\{a_0, b_0\}$ et celle exprimant le fait que $\{a_0\} \rightarrow \{b_0\}$ est infréquente en respectant le seuil maximum de support λ :

$$r_i \leftrightarrow (p_i^{a_0} \wedge p_i^{b_0}) \ , \ \sum_{i=1}^{m} r_i \leqslant \lambda \tag{5.10}$$

Enfin, nous introduisons la formule exprimant que $\{a_0\} \rightarrow \{b_0\}$ est infréquente en respectant le seuil maximum de support λ :

$$\sum_{i=1}^{m} r_i \leqslant m \times \lambda \tag{5.11}$$

Dans la définition 5.1, les items a_0 et b_0 sont interchangeables menant à des règles d'association indirectes symétriques. Pour éviter l'énumération de ces règles redondantes, nous imposons un ordre lexicographique sur l'ensemble des items et nous cassons les symétries entre a_0 et b_0 en ajoutant les contraintes $a_0 < b_0$ sur l'ensemble des items Ω exprimées comme suit :

$$\bigwedge_{a, a' \in \Omega, a' \leq a} \neg x_a^{a_0} \vee \neg x_{a'}^{b_0} \tag{5.12}$$

On utilise $\mathcal{E}_{IR}(\mathcal{D}, \lambda, \alpha, \beta)$ pour désigner l'encodage du problème de fouille de règles d'association indirectes (5.8) $\wedge$ (5.10) $\wedge$ (5.12).

Proposition 5.4. *Soient $\mathcal{D}$ une base de données transactionnelle, α un seuil minimum de support, β une seuil minimum de confiance, λ un seuil maximum de support et I une interprétation Booléenne de $\mathcal{E}_{IR}(\mathcal{D}, \lambda, \alpha, \beta)$. Alors, I est un modèle de $\mathcal{E}_{IR}(\mathcal{D}, \lambda, \alpha, \beta)$ ssi $\{a \in \Omega | I(x_a) = 1\} \rightarrow \{a \in \Omega | I(y_a) = 1\}$ est une règle d'association indirecte par rapport à α , β et λ.*

La preuve de la proposition 5.4 découle naturellement des différentes contraintes associées et de la proposition 5.3.

5.2.2 Règles d'association fermées

Une règle $X \rightarrow Y$ est *fermée* si elle satisfait en plus des contraintes de support minimum et de confiance minimum la contrainte que l'itemset $X \cup Y$ doit être fermé. Intuitivement, nous obtenons les règles d'association fermées en maximisant l'union de l'antécédent et la conséquence, sans diminuer ni le support ni la confiance. L'encodage SAT du problème de fouille de règles d'association fermées peut être obtenu simplement en ajoutant à l'encodage décrit précédemment ($\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$) la formule suivante :

$$\bigwedge_{a \in \Omega} (\bigwedge_{i=1}^{m} (q_i \rightarrow a \in I_i) \wedge \neg x_a \rightarrow y_a) \tag{5.13}$$

La formule (5.13) signifie que si on a $\mathcal{C}(X \cup Y, \mathcal{D}) = \mathcal{C}(X \cup Y \cup \{a\}, \mathcal{D})$ alors $a \in Y$. Comme une conséquence, si $X \rightarrow Y$ et $X \rightarrow Y \uplus \{a\}$ sont deux règles d'association valides, alors l'interprétation booléenne qui correspond à la règle $X \rightarrow Y$ est un contre-modèle de (5.13) ($\uplus$ correspond à l'union disjointe). De plus, s'il n y a pas d'item a tel que $X \rightarrow Y \uplus \{a\}$ est une règle d'association valide, alors l'interprétation booléenne correspondant à $X \rightarrow Y$ est un modèle de (5.13). En outre, il faut noter que la formule (5.13) encode que l'itemset $X \cup Y$ est fermé. On peut aussi noter que le nombre de clauses ajoutées par la formule (5.13) est égal au nombre d'items.

Nous utilisons $\mathcal{E}_{CAR}(\mathcal{D}, \alpha, \beta)$ pour désigne la conjonction des formules $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ et (5.13).

Proposition 5.5. *Soient $\mathcal{D}$ une base de données transactionnelles, α un seuil minimum de support, β une seuil minimum de confiance et $\mathcal{I}$ une interprétation Booléenne de $\mathcal{E}_{CAR}(\mathcal{D}, \alpha, \beta)$. Alors, $\mathcal{I}$ est un modèle de $\mathcal{E}_{CAR}(\mathcal{D}, \alpha, \beta)$ ssi $\{a \in \Omega | \mathcal{I}(x_a) = 1\} \rightarrow \{a \in \Omega | \mathcal{I}(y_a) = 1\}$ est une règle d'association fermée.*

Démonstration. Partie $\Rightarrow$. En utilisant la proposition 5.3, nous savons que $r : X = \{a \in \Omega | \mathcal{I}(x_a) = 1\} \rightarrow Y = \{a \in \Omega | \mathcal{I}(y_a) = 1\}$ est une règle d'association valide et sa couverture est $C(r, \mathcal{D}) = \{i \in \{1, \ldots, m\} | \mathcal{I}(q_i) = 1\}$. Supposons que r n'est pas fermée. Alors, il existe $a \notin X \cup Y$ tel que $C(r, \mathcal{D}) = C(X \cup Y \cup \{a\}, \mathcal{D}$. En utilisant la formule (5.13), $\mathcal{I}(y_a) = 1$ est vrai car $\mathcal{I}((\bigwedge_{i=1}^{m} q_i \rightarrow a \in I_i) \wedge \neg x_a) = 1$. En conséquence, nous avons une contradiction ($a \in X \cup Y$). En effet, r est une règle fermée.
Partie $\Leftarrow$. Soit $r : X \rightarrow Y$ une règle d'association fermée. L'interprétation Booléenne $\mathcal{I}$ est définie comme suit : $\forall a \in \Omega, \mathcal{I}(x_a) = 1$ ssi $a \in X$, $\mathcal{I}(y_a) = 1$ ssi $a \in Y$ et $\forall i \in \{1, \ldots, m\}, \mathcal{I}(p_i) = 1$ ssi $i \in C(X, \mathcal{D})$ et $\mathcal{I}(q_i) = 1$ ssi $i \in C(r, \mathcal{D})$. En utilisant la proposition 5.3, $\mathcal{I}$ est un modèle de $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$. Soit a un item tel que $\mathcal{I}((\bigwedge_{i=1}^{m} q_i \rightarrow a \in I_i) \wedge \neg x_a) = 1$ c-à-dire que nous avons $C(r, \mathcal{D}) = C(X \cup Y \cup \{a\}, \mathcal{D})$. En effet, nous aurons $a \in Y$ car $X \cup Y$ est un itemset fermé. En conséquence, $\mathcal{I}(y_a) = 1$ doit être vrai. En conséquence, $\mathcal{I}$ est aussi un modèle de la formule (5.13). □

Notons qu'en utilisant l'équivalence $A \rightarrow B \equiv \neg A \vee B$, la formule (5.13) est simplement reformulée comme une conjonction de clauses (CNF) comme suit :

$$\bigwedge_{a \in \Omega} \left(x_a \vee y_a \vee \bigvee_{i=1}^{m} (a \notin I_i \wedge q_i) \right)$$

En utilisant la contrainte de fermeture, $|\Omega|$ clauses sont ajoutées à $\mathcal{E}_{AR}(\mathcal{D}, \alpha, \beta)$ pour énumérer les règles d'association fermées.

Exemple 5.7. *Soit $\mathcal{D}$ la base de données transactionnelles représentée par la Table 5.1, la formule 5.13 correspond aux clauses suivantes pour capturer la fermeture :*

Name	Asso. Rules	Support	Confidence
r_1	$\{a\} \rightarrow \{b\}$	3/6	1
r_2	$\{a\} \rightarrow \{b, c, d\}$	3/6	1
r_3	$\{c\} \rightarrow \{d\}$	5/6	5/6
r_4	$\{c, d\} \rightarrow \{e, f, g\}$	2/6	2/5

TABLE 5.2 – Règles d'ssociation de la table 5.1

$(q_1 \vee q_2 \vee q_6 \vee x_a \vee y_a)$
$(q_1 \vee q_2 \vee q_6 \vee x_b \vee y_b)$
$(x_c \vee y_c)$
$(q_1 \vee q_6 \vee x_d \vee y_d)$
$(q_3 \vee q_4 \vee q_5 \vee x_e \vee y_e)$
$(q_3 \vee q_5 \vee q_6 \vee x_f \vee y_f)$
$(q_3 \vee q_4 \vee q_5 \vee q_6 \vee x_g \vee y_g)$

5.2.3 Règles d'association minimales non redondantes

Dans cette section, nous présentons notre encodage du problème d'extraction des règles non redondantes vers la satisfiabilité propositionnelle. Nous nous concentrons sur la représentation intéressante qui correspond au règles d'association appelées minimales non redondantes (MNRs) [Kry98b, BPT+00].

Avant de donner l'encodage SAT de ce problème, rappelons qu'une règle d'association $r : X \rightarrow Y$ dans une base de données transactionnelles $\mathcal{D}$ est minimale non redondante si et seulement s'il n'y a pas de règle d'association $r' : X' \rightarrow Y'$ différente de r qui satisfait les propriétés suivantes :

1. le support de r dans $\mathcal{D}$ est égal au support de r' dans $\mathcal{D}$, c-à-d, $S(r, \mathcal{D}) = S(r', \mathcal{D})$;
2. la confiance de r dans $\mathcal{D}$ est égale à la confiance de r' dans $\mathcal{D}$, c-à-d, $Conf(r, \mathcal{D}) = Conf(r', \mathcal{D})$;
3. $X' \subseteq X$ et $Y \subseteq Y'$.

En d'autres termes pour une classe d'équivalence de règles d'associations qui ont la même couverture et la même confiance, les règles minimales non redondantes sont celles qui ont les plus petits antécédents et les plus grandes conséquences.

Exemple 5.8. *Considérons les règles d'association données dans la table 5.2. Dans cet ensemble de règles,* $r_2 : \{a\} \rightarrow \{b, c, d\}$ *est une règle minimal non redondante tandis que* $r_1 : \{a\} \rightarrow \{b\}$ *ne l'est pas car* r_1 *et* r_2 *ont le même support et la même confiance et que* $\{b\} \subset \{b, c, d\}$.

Dans la proposition 5.6, nous soulignons que toutes les règles d'association minimales non redondantes sont fermées.

Proposition 5.6. *Si* $r : X \rightarrow Y$ *est une règle minimale non redondante dans une base de données transactionnelles* $\mathcal{D}$ *alors* $X \cup Y$ *est un itemset fermé dans* $\mathcal{D}$.

Démonstration. Supposons que $X \cup Y$ n'est pas un itemset fermé. Alors, il existe un item $a \notin X \cup Y$ tel que $S(X \cup Y, \mathcal{D}) = S(X \cup Y \cup \{a\}, \mathcal{D})$. Considérons maintenant la règle $r' : X \rightarrow Y \cup \{a\}$. Nous obtenons clairement $S(r) = S(r')$ et $Conf(r) = Conf(r')$ puisque $S(X \cup Y, \mathcal{D}) = S(X \cup Y \cup \{a\}, \mathcal{D})$. Ainsi, r n'est pas une règle minimale non redondante et nous obtenons une contradiction.

□

Autrement dit, les règles d'association minimales non redondantes sont des règles fermées dans lesquelles les antécédents sont minimaux par rapport à l'inclusion ensembliste. En utilisant cette propriété, les auteurs de [BPT+00] ont proposé une caractérisation des antécédents des règles minimales non redondantes, appelées générateurs minimaux.

Rappelons qu'étant donné un itemset fermé X dans une base de données transactionnelles $\mathcal{D}$, un générateur minimal de X, noté $X' \subseteq X$ est un itemset qui vérifie :

— $S(X', \mathcal{D}) = S(X, \mathcal{D})$;

— il n'y a pas d'itemset $X'' \subseteq X$ tel que $X'' \subset X'$ et $S(X'', \mathcal{D}) = S(X, \mathcal{D})$.

Les algorithmes usuels utilisent l'ensemble des itemsets fréquents et fermés avec les générateurs minimaux pour extraire l'ensemble des règles minimales non redondantes. Alors, les approches existantes pour la fouille des règles minimales procèdent en deux étapes. Dans notre approche, nous proposons d'étendre l'encodage SAT proposé $\mathcal{E}_{CAR}(\mathcal{D}, \alpha, \beta)$ pour extraire les règles d'association minimales non redondantes en une seule étape.

Afin de définir un encodage SAT du problème d'énumération des règles d'association minimales non redondantes, il suffit d'étendre l'encodage $\mathcal{E}_{CAR}(\mathcal{D}, \alpha, \beta)$ décrit précédemment avec la formule qui oblige chaque antécédent à être un générateur minimal. En effet, nous utilisons une formule qui représente le fait que si $S(X \rightarrow Y, \mathcal{D}) = S(X \setminus \{a\} \rightarrow Y, \mathcal{D})$, alors a doit être exclus de X, c-à-d, $a \notin X$. Cependant, nous écrivons la contraposition de cette propriété. En effet, la formule (5.14) exprime que pour tout item a, si a appartient à X, alors le support de X est inférieur à celui de $X \setminus \{a\}$.

$$(\bigwedge_{a \in \Omega} x_a \rightarrow \bigvee_{(i \in 1..m,\ a \notin I_i)} (\bigwedge_{b \notin I_i \cup \{a\}} \neg x_b)) \vee (\sum_{b \in \Omega} x_b = 1) \tag{5.14}$$

L'intuition de cet encodage est qu'un item a est dans X si et seulement s'il existe au moins une transaction i falsifiée uniquement par a c-à-dire la transaction i contient tous les items de X sauf a. Par conséquent, on ne peut pas supprimer a de X car si on le supprime la transaction i devient $vraie$ et le support de X change.

On utilise $\mathcal{E}_{MNR}(\mathcal{D}, minsupp, minconf)$ pour noter l'encodage (5.1) $\wedge$ (5.2) $\wedge$ (5.3) $\wedge$ (5.4) $\wedge$ (5.5) $\wedge$ (5.7) $\wedge$ (5.13) $\wedge$ (5.14).

La correction de $\mathcal{E}_{MNR}(\mathcal{D}, minsupp, minconf)$ vient directement de la proposition suivante :

Proposition 5.7. *La règle d'association $r : X \rightarrow Y$ est une règle minimale non redondante si et seulement si r est une règle d'association fermée, et $|X| = 1$ ou, pour tout item $a \in X$, $S(X, \mathcal{D}) < S(X \setminus \{a\}, \mathcal{D})$.*

Démonstration.
Partie $\Rightarrow$. En utilisant la proposition 5.6, nous savons que r est une règle d'association fermée. Supposons qu'il existe un item $a \in X$ tel que $S(X, \mathcal{D}) = S(X \setminus \{a\}, \mathcal{D})$. Alors, $r' : X \setminus \{a\} \rightarrow Y \cup \{a\}$ est une règle d'association fermée telle que $S(r, \mathcal{D}) = S(r', \mathcal{D})$ et $Conf(r, \mathcal{D}) = Conf(r', \mathcal{D})$. Ainsi, nous avons une contradiction puisque r est une règle d'association minimale non redondante.
Partie $\Leftarrow$. En utilisant le fait que r est une règle d'association fermée, nous savons qu'il n'y a pas de règle d'association $r' : X' \rightarrow Y'$ telle que $X \cup Y \subset X' \cup Y'$ et $S(r, \mathcal{D}) = S(r', \mathcal{D})$. De plus, sachant que $S(X, \mathcal{D}) < S(X \setminus \{a\}, \mathcal{D})$ pour chaque $a \in X$, nous obtenons $Conf(X \setminus \{a\} \rightarrow Y \cup \{a\}, \mathcal{D}) < Conf(r, \mathcal{D})$ pour chaque $a \in X$. En conséquence, r est une règle d'association non redondante. □

La correction de notre encodage signifie qu'une interprétation Booléenne $\mathcal{I}$ est un modèle de $\mathcal{E}_{MNR}(\mathcal{D}, minsupp, minconf)$ si et seulement si $X = \{a \in \Omega \mid \mathcal{I}(x_a) = 1\} \rightarrow Y = \{b \in \Omega \mid \mathcal{I}(y_b) = 1\}$ est une règle d'association minimale non redondante.

Proposition 5.8. *L'encodage $\mathcal{E}_{MNR}(\mathcal{D}, minsupp, minconf)$ est correct.*

Démonstration. ça provient de la correction de l'encodage $\mathcal{E}_{CAR}(\mathcal{D}, minsupp, minconf) \equiv (5.1) \wedge (5.2) \wedge (5.3) \wedge (5.4) \wedge (5.5) \wedge (5.7) \wedge (5.13)$ en respectant le problème de génération des règles d'association fermées, et la proposition 5.7, et le fait que (5.14) exprime que $Supp(X, \mathcal{D}) < Supp(X \setminus \{a\}, \mathcal{D})$ pour chaque $a \in X$.

□

Notons que la contrainte (5.14) n'est pas une formule CNF. Afin d'éviter l'explosion en terme de nombre de clauses résultantes de la transformation de (5.14) en CNF, de nouvelles variables peuvent être ajoutées pour représenter les sous formules de la forme $(\bigwedge_{b \notin I_i \cup \{a\}} \neg x_b)$, c-à-d, $z_i \leftrightarrow (\bigwedge_{b \notin I_i \cup \{a\}} \neg x_b)$. Toutefois, en utilisant cette transformation, le nombre de clauses résultantes est en $O(m \times |\Omega|^2)$ ce qui peut rendre l'énumération des modèles beaucoup plus difficile. Pour limiter le nombre de clauses, nous proposons la transformation exprimée dans la formule (5.15) qui est équivalente à la propriété capturée par (5.14).

$$(\bigwedge_{a \in \Omega} (x_a \rightarrow \bigvee_{(i \in 1..m,\ a \notin I_i)} z_i)) \wedge (\bigwedge_{i \in 1..m} (z_i \rightarrow \sum_{b \notin I_i} x_b \leq 1)) \vee (\sum_{b \in \Omega} x_b = 1) \tag{5.15}$$

En effet, cette transformation vient du fait que $(\bigwedge_{b \notin I_i \cup \{a\}} \neg x_b)$ est équivalente à $(\sum_{b \notin I_i} x_b \leq 1)$ dans le cas où I_i ne contient pas a. En conséquence, (5.15) exprime exactement les exigences de (5.14). Les variables additionnelles z_i permettent d'obtenir un encodage efficace. L'intuition derrière cet encodage est d'abords de détecter les transactions qui sont falsifiées par au plus un item en utilisant des contraintes AtMostOne. Ensuite, forcer chaque item dans X à être le seul à falsifier au moins une transaction avec une disjonction entre les variables z_i.

Exemple 5.9. *Soit $\mathcal{D}$ la base de données transactionnelles représentée par la Table 5.1, la formule 5.15 correspond aux clauses suivantes pour capturer la minimalité de X de la règle $X \rightarrow Y$:*

$y \rightarrow (x_a + x_b + x_c + x_d + x_e + x_f + x_g \geq 1)$	$(\neg x_a \vee z_1 \vee z_2 \vee z_6 \vee \neg y)$
$z_1 \rightarrow (x_a + x_b + x_d \leqslant 1)$	$(\neg x_b \vee z_1 \vee z_2 \vee z_6 \vee \neg y)$
$z_2 \rightarrow (x_a + x_b \leqslant 1)$	$(\neg x_c \vee \neg y)$
$z_3 \rightarrow (x_e + x_f + x_g \leqslant 1)$	$(\neg x_d \vee z_1 \vee z_6 \vee \neg y)$
$z_4 \rightarrow (x_e + x_g \leqslant 1)$	$(\neg x_e \vee z_3 \vee z_4 \vee z_5 \vee \neg y)$
$z_5 \rightarrow (x_e + x_f + x_g \leqslant 1)$	$(\neg x_f \vee z_3 \vee z_5 \vee z_6 \vee \neg y)$
$z_6 \rightarrow (x_a + x_b + x_d + x_f + x_g \leqslant 1)$	$(\neg x_a \vee z_3 \vee z_4 \vee z_5 \vee z_6 \vee \neg y)$

Notons que la formule (5.15) peut être encodée en $O(m \times |\Omega|)$ clauses plutôt que $O(m \times |\Omega|^2)$ dans la formulation précédente. Une contrainte linéaire de la forme $\sum_{i=1}^{n} x_i \leq 1$ peut être encodée linéairement [Sin05] en utilisant des variables additionnelles comme le montre la formule (5.16) (voir chapitre 1).

$$(\neg x_1 \vee s_1) \wedge (\neg x_n \vee \neg s_{n-1}) \wedge \bigwedge_{1<i<n} (\neg x_i \vee s_i) \wedge (\neg s_{i-1} \vee s_i) \wedge (\neg x_i \vee \neg s_{i-1}) \tag{5.16}$$

$$(\neg x_1 \vee s_1) \wedge (\mathbf{y} \vee \neg x_n \vee \neg s_{n-1}) \wedge \bigwedge_{1<i<n} (\neg x_i \vee s_i) \wedge (\neg s_{i-1} \vee s_i) \wedge (\mathbf{y} \vee \neg x_i \vee \neg s_{i-1}) \tag{5.17}$$

Ainsi, la contrainte $(\neg y \rightarrow \sum_{i=1}^{n} x_i \leq 1)$ est encodée en rajoutant y à chaque clause de (5.16). Cependant, cela peut ralentir le processus de la propagation unitaire. En effet, lorsque plus qu'une variable x_i est affectée à $vrai$, y ne se déduit pas pour être vrai directement par propagation unitaire. Pour

augmenter la puissance de la propagation unitaire, il faut ajouter y uniquement au clauses binaires négatives de (5.16) comme le montre la formule (5.17). Plus d'explications sont présentées dans la section 5.3.

Il convient de noter qu'on peut utiliser certaines des contraintes ci-dessus pour énumérer tous les générateurs minimaux. Comme mentionné précédemment, les générateurs minimaux sont les antécédents des règles minimales non redondantes. En conséquence, l'encodage (5.3) $\wedge$ (5.5) $\wedge$ (5.15) (limité à X) permet d'avoir l'ensemble des générateurs minimaux qui correspond à l'ensemble des itemsets minimaux (aussi appelés itemsets libres) .

5.2.4 Règles d'association avec un minimum de conditions et un minimum de conséquences

Une autre notion de règles non redondantes a été définie dans le travail de M. Zaki [Zak04]. C'est légèrement différent des règles représentatives définies dans [Kry98a]. Ce problème consiste à extraire des règles d'association, appelées les règles les plus générales (MGR), qui ont le plus petit antécédent et la plus petite conséquence (en termes d'inclusion) dans une classe d'équivalence de règles (avec la même couverture et la même confiance)

Définition 5.2. *[Zak04] Une règle d'association $r : X \rightarrow Y$ est une règle non redondante dans une base de données transactionnelles $\mathcal{D}$ ssi il n'y a pas de règle $r' : X' \rightarrow Y'$ dans $\mathcal{D}$ différente de r tels que (i) $S(r, \mathcal{D}) = S(r', \mathcal{D})$, (ii) $Conf(r, \mathcal{D}) = Conf(r', \mathcal{D})$ et (iii) $X' \subseteq X$ et $Y' \subseteq Y$.*

Contrairement à la notion de non redondance présentée dans la section 5.2.3, la contrainte de fermeture sur $X \cup Y$ est évidemment omise dans la définition de Zaki.

Exemple 5.10. *Considérant à nouveau les règles d'association de la Table 5.2. $r_2 : \{a\} \rightarrow \{b, c, d\}$ est redondante tandis que $r_1 : \{a\} \rightarrow \{b\}$ ne l'est pas.*

La proposition 5.9 donne une caractérisation des règles d'association non redondantes de Zaki.

Proposition 5.9. *Etant donnée une règle d'association $r : X \rightarrow Y$ dans une base de données transactionnelles $\mathcal{D}$, r est une règle non redondante ssi (i) $|X| = 1$ ou $\forall a \in X$, $S(X \setminus \{a\}, \mathcal{D}) > S(X, \mathcal{D})$; et (ii) $|Y| = 1$ ou $\forall b \in Y$, $S(X \cup Y) < S(X \cup Y \setminus \{b\})$.*

Démonstration.
Partie $\Rightarrow$. Supposons que $|X| > 1$ et qu'il existe $a \in X$ tel que $S(X \setminus \{a\}, \mathcal{D}) = S(X, \mathcal{D})$. Alors, $S(X \setminus \{a\} \rightarrow Y, \mathcal{D}) = S(r, \mathcal{D})$. De plus, nous avons $S(X \cup Y \setminus \{a\}, \mathcal{D}) = S(X \cup Y, \mathcal{D})$. Ainsi, nous avons $Conf(X \setminus \{a\} \rightarrow Y, \mathcal{D}) = Conf(r, \mathcal{D})$. En conséquence, nous avons une contradiction puisque r est une règle non redondante et nous obtenons la propriété (i).

Supposons maintenant qu'il existe $b \in Y$ tel que $|Y| > 1$ et $S(X \cup Y \setminus \{b\}, \mathcal{D}) = S(X \cup Y, \mathcal{D})$. Alors, $Conf(X \rightarrow Y \setminus \{b\}, \mathcal{D}) = Conf(X \rightarrow Y, \mathcal{D})$. De plus, nous avons $S(X \rightarrow Y \setminus \{b\}, \mathcal{D}) = S(X \rightarrow Y, \mathcal{D})$. Ainsi, en utilisant le fait que r est une règle non redondante, nous avons une contradiction et nous obtenons la propriété (ii). *Partie* $\Leftarrow$. Supposons que r est une règle non redondante. Alors , il existe $a \in X \cup Y$ tel que $S(X \setminus \{a\} \rightarrow Y, \mathcal{D}) = S(r, \mathcal{D})$ si $a \in x$, et $Conf(X \rightarrow Y \setminus \{a\}, \mathcal{D}) = conf(r, \mathcal{D})$ sinon. Ainsi, nous avons $S(X \setminus \{a\}, \mathcal{D}) = S(X, \mathcal{D})$ si $a \in X$, et $S(X \cup Y) = S(X \cup Y \setminus \{b\})$ sinon. En conséquence, en utilisant les propriétés (i) et (ii) nous avons une contradiction. Donc, r est non redondante.

□

En utilisant la caractérisation donnée par la Proposition 5.9, il suffit d'ajouter à l'encodage $\mathcal{E}_{MNR}(\mathcal{D}, minsupp, minconf)$, sans contrainte de fermeture, la nouvelle contrainte (5.18) qui représente la propriété (ii) pour avoir un encodage qui correspond au règles non redondantes de Zaki.

$$\bigwedge_{a\in\Omega} y_a \rightarrow \Big(\bigvee_{(i\in 1..m,\ a\notin I_i)} (p_i \wedge \bigwedge_{b\notin I_i\cup\{a\}} \neg y_b) \Big) \vee \Big(\sum_{b\in\Omega} y_b = 1\Big) \tag{5.18}$$

Il convient de noter que la contrainte (5.18) est très similaire à (5.14). La différence réside dans le fait que nous utilisons les variables p_i pour raisonner sur la couverture de $X \cup Y$ et non pas Y. En outre, on peut facilement voir que (5.18) peut être encodée en une formule CNF de la même façon que (5.14).

5.3 Encodages de contraintes de cardinalité conditionnelles

Dans cette section, nous montrons comment les encodages de contraintes de cardinalité de la section 1.2.3 peuvent être étendues efficacement pour encoder les contraintes de cardinalité conditionnelle de la forme $y \rightarrow \sum_{i=1}^{n} x_i \leqslant k$ que nous avons utilisée dans l'encodage des règles d'associations tout en préservant la cohérence d'arc généralisée maintenue par la propagation unitaire. Plus précisément, pour une telle contrainte de cardinalité conditionnelle, le maintien de GAC signifie que lorsque y est assigné à vrai, l'encodage doit maintenir GAC sur la contrainte de cardinalité $\sum_{i=1}^{n} x_i \leqslant k$. D'un autre côté, lorsque la contrainte de cardinalité est falsifiée par l'assignation courante, la variable $\neg y$ doit être déduite par propagation unitaire.

Une observation importante qui peut être faite à partir des encodages basés sur SAT de la contrainte de cardinalité présentés dans la section 1.2.3 du premier chapitre, est que la formule obtenue est une formule de Horn. Introduisons d'abord une propriété importante, permettant de saisir l'intuition derrière les encodages que nous proposons dans cette section.
Soit $\mathcal{F}$ une formule de Horn, la sous-formule $\mathcal{F}^-$ désigne l'ensemble des clauses négatives de $\mathcal{F}$ et $\mathcal{F}^+$ l'ensemble des clauses de $\mathcal{F}$ contenant exactement un littéral positif. Nous utilisons dans la suite de cette section la notation $\mathcal{F} \models^* c$ pour noter que la clause c est déduite par propagation unitaire à partir de $\mathcal{F}$, c-à-d, $(\mathcal{F} \wedge \bar{c})^* \models \bot$.

Proposition 5.10. *Soit $\mathcal{F}$ une formule de Horn et $\mathcal{I} = \{x_1, x_2, \ldots, x_k\} \subseteq \mathcal{P}(\mathcal{F})$ une interprétation. $\mathcal{F}|_{\mathcal{I}} \models^* \bot$ ssi $\exists c \in \mathcal{F}^-$ tel que $\mathcal{F}^+|_{\mathcal{I}} \models^* \bar{c}$.*

Démonstration. $(\Rightarrow)$. Considérons la formule $\mathcal{F}^+|_{\mathcal{I}}$. Supposons qu'il n'y a pas de clause $c \in \mathcal{F}^-$ telle que $\mathcal{F}^+|_{\mathcal{I}} \models^* \bar{c}$. Soit S l'ensemble des littéraux unitaires de $\mathcal{F}^+|_{\mathcal{I}}$ y compris les littéraux de $\mathcal{I}$. Nous pouvons noter que de $\mathcal{F}^+|_{\mathcal{I}}$ seuls des littéraux unitaires positifs supplémentaires peuvent être déduits par propagation unitaire ($S \supseteq \mathcal{I}$). Donc S est un ensemble de littéraux positifs. $S \cup \overline{(\mathcal{P}(\mathcal{F}) \setminus S)}$ est clairement un modèle de $\mathcal{F}|_{\mathcal{I}}$. En effet, chaque clause de $\mathcal{F}^+|_{\mathcal{I}}$ est une clause satisfaite (son littéral positif est dans S) ou elle contient au moins un littéral négatif. En effet, propager des littéraux positifs sur $\mathcal{F}^+$ conduit à une formule où les clauses restantes contiennent un littéral positif et au moins un littéral négatif. Les clauses restantes de $\mathcal{F}^- \wedge S$ sont des clauses négatives avant de supprimer de chaque clause les littéraux de $\bar{S}$. Ensuite, en affectant les variables restantes $\mathcal{P}(\mathcal{F}) \setminus S$ à $faux$, nous obtenons un modèle $\mathcal{I}' = S \cup \overline{(V(\mathcal{F}) \setminus S)}$ de la formule $\mathcal{F}$. Comme $\mathcal{I} \subseteq \mathcal{I}'$, cela contredit l'hypothèse que $\mathcal{F}|_{\mathcal{I}}$ est insatisfiable.
$(\Leftarrow)$. De $\mathcal{F}^+|_{\mathcal{I}} \models^* \bar{c}$, nous avons $(\mathcal{F}^+ \wedge c)|_{\mathcal{I}} \models^* \bot$. Comme $\mathcal{F}^+ \wedge c \subseteq \mathcal{F}$, alors $\mathcal{F}|_{\mathcal{I}} \models^* \bot$. □

Étant donné une formule de Horn $\mathcal{F}$, la proposition 5.10 exprime que l'insatisfiabilité sous toute interprétation fait d'un ensemble de littéraux positifs, est causée par une clause de $\mathcal{F}^-$. Comme la contrainte de cardinalité est habituellement encodée comme une formule de Horn $\mathcal{F}$, pour maintenir GAC sur l'encodage de $y \rightarrow \mathcal{F}$, il suffit d'ajouter disjonctivement $\neg y$ à $\mathcal{F}^-$.

5.3.1 Encodages de contraintes AtMostOne conditionnelles

Considérons d'abord la contrainte AtMostOne conditionnelle $y \rightarrow \sum_{i=1}^{n} x_i \leqslant 1$. De nombreux encodages ont été proposés pour transformer la contraint AtMostOne en une formule CNF. Considérons deux encodages standard de cette contrainte.

Encodage par paire d'AtMostOne (Pairwise encoding) : L'encodage par paire classique peut être obtenu en considérant l'ensemble de toutes les clauses binaires négatives construites sur l'ensemble des variables $\{x_1, \ldots, x_n\}$ (voir section 1.2.3).

$$\bigwedge_{1 \leq i < j \leq n} (\neg \mathbf{y} \vee \neg x_i \vee \neg x_j) \tag{5.19}$$

La formule (5.19) encodant la contrainte AtMostOne conditionnelle $y \rightarrow \sum_{i=1}^{n} x_i \leqslant 1$ est obtenu en ajoutant simplement $\neg y$ à toutes les clauses de la formule CNF obtenues de l'encodage par paire. Il est simple de remarquer que la formule obtenue (5.19) permet de maintenir la cohérence d'arc généralisée. En effet, toute assignation de deux littéraux x_i et x_j à $vrai$, permet de déduire $\neg y$ par propagation unitaire. D'un autre côté, si y est assigné à $vrai$, la contrainte conditionnelle est réduite à une simple contrainte AtMostOne qui préserve la cohérence d'arc généralisée.

Encodage basé sur un compteur séquentiel de la contrainte AtMostOne conditionnelle : Avec l'encodage basé sur le compteur séquentiel, en ajoutant $\neg y$ à toutes les clauses de l'encodage de base, nous obtenons une nouvelle formulation de la contrainte AtMostOne conditionnelle (formule (5.20)) qui ne maintient pas la cohérence d'arc généralisée.

$$\begin{array}{l}(\neg \mathbf{y} \vee \neg x_1 \vee p_1) \wedge (\neg \mathbf{y} \vee \neg x_n \vee \neg p_{n-1}) \wedge \\ \bigwedge_{1 < i < n} (\neg \mathbf{y} \vee \neg x_i \vee p_i) \wedge (\neg \mathbf{y} \vee \neg p_{i-1} \vee p_i) \wedge (\neg \mathbf{y} \vee \neg x_i \vee \neg p_{i-1})\end{array} \tag{5.20}$$

En effet, assigner deux littéraux de $\{x_1, \ldots, x_n\}$ ne nous permet pas de déduire $\neg y$ par propagation unitaire. Par exemple, en affectant x_1 et x_n à $vrai$, les deux premières clauses de (5.20) deviennent binaires.

Pour maintenir la cohérence d'arc généralisée pour la contrainte AtMostOne conditionnelle en utilisant un compteur séquentiel, $\neg y$ doit être ajouté à un sous-ensemble des clauses comme représenté dans la formule (5.21).

$$\begin{array}{l}(\neg x_1 \vee p_1) \wedge (\neg \mathbf{y} \vee \neg x_n \vee \neg p_{n-1}) \wedge \\ \bigwedge_{1 < i < n} (\neg x_i \vee p_i) \wedge (\neg p_{i-1} \vee p_i) \wedge (\neg \mathbf{y} \vee \neg x_i \vee \neg p_{i-1})\end{array} \tag{5.21}$$

Proposition 5.11. *La formule CNF (5.21) encodant la contrainte $y \rightarrow \sum_{i=1}^{n} x_i \leqslant 1$ en utilisant un compteur séquentiel maintient la cohérence d'arc généralisée par propagation unitaire.*

Démonstration. La preuve de cette proposition est une conséquence directe de la proposition 5.10. En effet, l'encodage de $\sum_{i=1}^{n} x_i \leqslant 1$ est une formule de Horn. Par conséquent, lorsque plus d'un littéral de $\{x_1, \ldots, x_n\}$ sont affectés à $vrai$, une clause provenant des clauses négatives de l'encodage devient fausse. Par conséquent, pour coder $y \rightarrow \sum_{i=1}^{n} x_i \leqslant 1$, il suffit d'ajouter $\neg y$ aux deux clauses négatives comme indiqué dans la contrainte (5.21). En effet, supposons que nous assignions deux variables arbitraires x_i et x_j (avec $1 < i < j < n$) à $vrai$. De l'affectation de x_i à $vrai$ et de la clause $(\neg x_i \vee p_i)$, nous déduisons le littéral unitaire p_i. Ensuite, à partir de la clause $(\neg p_i \vee p_{i+1})$, nous déduisons un autre littéral unitaire p_{i+1}. Cette chaîne de littéraux unitaires propagés se poursuit jusqu'à p_{j-1}. Maintenant, si nous assignons x_j à $vrai$, la clause $(\neg y \vee \neg x_j \vee \neg p_{j-1})$ nous permet de déduire $\neg y$ car p_{j-1} (littéral unitaire propagé) et x_j sont assignés à $vrai$. Considérons un autre cas où x_1 est affecté à $vrai$. Une telle affectation nous permet de déduire , grâce à la propagation unitaire, les littéraux $p_1, \ldots, p_{n-1}$. Puis en assignant tout autre littéral x_j (avec $j \neq 1$), nous déduisons le littéral $\neg y$ grâce à la clause $(\neg y \vee \neg x_j \vee \neg p_{j-1})$. Évidemment, l'affectation de y à $vrai$ conduit à l'encodage classique de la contrainte AtMostOne basé sur le compteur séquentiel qui préserve la cohérence d'arc généralisée par propagation unitaire. □

Encodage basé sur les réseaux de tri de l'AtMostOne conditionnel : L'encodage basé sur réseau de tri de la contrainte conditionnelle AtMostOne est similaire à la contrainte AtMostK conditionnelle décrite dans la section 5.3.2. Il est défini comme suit :

$$\Phi_{SN}^{n,1}(X; Z) \wedge (\neg \mathbf{y} \vee \neg z_2) \tag{5.22}$$

La proposition 5.13 montre que l'encodage maintient la cohérence d'arc généralisée par propagation unitaire pour toute valeur de $k > 0$.

5.3.2 Encodages de contraintes AtMostK conditionnelles

Considérons maintenant le cas général de la contrainte AtMostK conditionnelle.

Encodage basé sur le problème de pigeons : Dans la sous-section 1.2.3 du premier chapitre, nous avons présenté l'encodage basé sur le problème de pigeons de la contrainte de cardinalité AtLeastK de la forme $\sum_{i=1}^{n} x_i \geqslant k$ proposé dans [JSS14]. Pour des raisons de clarté et de cohérence, et comme la contrainte $\sum_{i=1}^{n} x_i \leqslant k$ peut être réécrit de manière équivalente en tant que contrainte AtLeastK $\sum_{i=1}^{n} \neg x_i \geqslant n - k$ pour l'encodage basé le problème de pigeons, nous considérons la contrainte AtLeastK conditionnelle $y \rightarrow \sum_{i=1}^{n} x_i \geqslant k$. Pour préserver GAC, $\neg y$ doit être ajouté à un sous-ensemble limité de clauses de l'encodage $ph\mathcal{P}_n^k$. Seules les clauses positives de l'encodage sont augmentées par $\neg y$.

$$\bigwedge_{1 \leq i \leq k} (\neg \mathbf{y} \bigvee_{1 \leq j \leq n-k+1} p_{ij}) \tag{5.23}$$

$$\bigwedge_{1 \leq i \leq k} \bigwedge_{1 \leq j \leq n-k+1} (x_{i+j-1} \vee \neg p_{ij}) \tag{5.24}$$

$$\bigwedge_{1 \leq i < k} \bigwedge_{1 \leq j < n-k+1} (\neg p_{(i+1)j} \vee \bigvee_{1 \leq l \leq j} p_{il}) \tag{5.25}$$

Exemple 5.11. *Considérons l'inéquation* $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geqslant 4$. *En utilisant l'encodage basé sur le problème des pigeons* $ph\mathcal{P}_6{}^4$, *on obtient la formule CNF suivante :*

$$
\begin{array}{llll}
p_{11} \vee \neg p_{21} & x_1 \vee \neg p_{11} & x_2 \vee \neg p_{12} & \\
p_{11} \vee p_{12} \vee \neg p_{22} & x_2 \vee \neg p_{21} & x_3 \vee \neg p_{13} & \neg \mathbf{y} \vee p_{11} \vee p_{12} \vee p_{13} \\
p_{21} \vee \neg p_{31} & x_3 \vee \neg p_{22} & x_3 \vee \neg p_{31} & \neg \mathbf{y} \vee p_{21} \vee p_{22} \vee p_{23} \\
p_{21} \vee p_{22} \vee \neg p_{32} & x_4 \vee \neg p_{23} & x_4 \vee \neg p_{32} & \neg \mathbf{y} \vee p_{31} \vee p_{32} \vee p_{33} \\
p_{31} \vee \neg p_{41} & x_4 \vee \neg p_{41} & x_5 \vee \neg p_{33} & \neg \mathbf{y} \vee p_{41} \vee p_{42} \vee p_{43} \\
p_{31} \vee p_{32} \vee \neg p_{42} & x_5 \vee \neg p_{42} & x_6 \vee \neg p_{43} &
\end{array}
$$

Proposition 5.12. *L'encodage (5.23) ∧ (5.24) ∧ (5.25) préserve la cohérence d'arc généralisée de* $y \rightarrow \sum_{i=1}^{n} x_i \geqslant k$.

Démonstration. Notons que l'encodage basé sur le problème de pigeons de la contrainte $\sum_{i=1}^{n} x_i \geqslant k$ est une formule reverse-Horn. Donc la proposition 5.10 peut être légèrement modifié pour être adapté au cas de reverse-Horn en considérant des affectations de variables à $faux$ et des clauses positives. En conséquence, on peut conclure que l'ajout de $\neg y$ aux clauses positives est suffisant pour maintenir la GAC par propagation unitaire. Schématisons maintenant la preuve en utilisant l'exemple 5.11 qui montre l'encodage de la contrainte $y \rightarrow x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geqslant 4$ en utilisant l'encodage basé sur les problème des pigeons. L'encodage CNF de la contrainte de cardinalité $y \rightarrow \sum_{i=1}^{6} x_i \geqslant 4$ est obtenu en ajoutant disjonctivement $\neg y$ aux clauses positives (clauses sur le côté droit). Montrons qu'en assignant trois variables parmi $\{x_1, \ldots, x_6\}$ à $faux$, nous déduisons $\neg y$ par propagation unitaire. Supposons que x_1, x_2 et x_3 sont assignés à $faux$. A partir des deuxième et troisième ensembles de clauses , nous déduisons par propagation unitaire $\neg p_{11}$, $\neg p_{12}$, $\neg p_{21}$, $\neg p_{13}$, $\neg p_{22}$ et $\neg p_{31}$. Par conséquent, à partir de la clause $(\neg \mathbf{y} \vee p_{11} \vee p_{12} \vee p_{13})$, nous déduisons $\neg y$. Considérons un autre cas, disons que x_2, x_4 et x_6 sont affectés à $faux$. Par propagation unitaire, nous déduisons $\neg p_{12}$, $\neg p_{21}$, $\neg p_{23}$, $\neg p_{32}$, $\neg p_{41}$, $\neg p_{43}$, $\neg p_{31}$, $\neg p_{41}$, $\neg p_{42}$. A partir de la clause $(\neg y \vee p_{41} \vee p_{42} \vee p_{43})$, nous déduisons $\neg y$. De la même façon, toute autre affectation de trois variables (ou plus) de l'ensemble $\{x_1, \ldots, x_6\}$ à $faux$ produit $\neg y$ par propagation unitaire.

□

Encodage basé sur les réseaux de tri : Considérons maintenant l'encodage basé sur les réseaux triés de la contrainte AtMostK conditionnelle $y \rightarrow \sum_{i=1}^{n} x_i \leqslant k$. En utilisant l'encodage basé sur les réseaux de triee de la contrainte AtMostK (voir la section 1.2.3), sa variante conditionnelle peut être représentée par $y \rightarrow (\Phi_{SN}^{n,k}(X; Z) \wedge \neg z_{k+1})$ ce qui est équivalent à la formule CNF $(\neg \mathbf{y} \vee \Phi_{SN}^{n,k}(X; Z)) \wedge (\neg \mathbf{y} \vee \neg z_{k+1})$. Comme indiqué à la section 1.2.3, le comparateur de base de deux variables propositionnelles, 2-$comp(x_i, x_j; z_i, z_j)$, est un bloc de base de l'encodage basé sur les réseaux tri $\Phi_{SN}^{n,k}(X; Z)$, c-à-d, une conjonction de plusieurs formules encodant chacune un 2-comparateur de base. Par conséquent, la formule conditionnelle $(\neg y \vee \Phi_{SN}^{n,k}(X; Z))$ peut être encodée en CNF en ajoutant $\neg y$ à toutes les clauses de chacun des 2-comparateurs de base, ce qui conduit à plusieurs comparateurs conditionnels de la forme $y \rightarrow$ 2-$comp(x_i, x_j; z_i, z_j)$, écrit sous une forme clausale comme :

$$(\neg \mathbf{y} \vee \neg x_i \vee z_i) \wedge (\neg \mathbf{y} \vee \neg x_j \vee z_i) \wedge (\neg \mathbf{y} \vee \neg x_i \vee \neg x_j \vee z_j) \tag{5.26}$$

Comme nous pouvons le voir, assigner un littéral d'entrée x_i ou x_j d'un 2-comparateur conditionnel ne permet pas de déduire aucun littéral par propagation unitaire car toutes les clauses de (5.26) deviennent

binaires. En effet, pour maintenir la cohérence d'arc généralisé pour la contrainte AtMostK en utilisant l'encodage basé sur les réseaux de tri, $\neg y$ doit être ajouté seulement à la clause unitaire $\neg z_{k+1}$:

$$\Phi_{SN}^{n,k}(X;Z) \wedge (\neg \mathbf{y} \vee \neg z_{k+1}) \tag{5.27}$$

Proposition 5.13. *L'encodage* $\Phi_{SN}^{n,k}(X;Z) \wedge (\neg \mathbf{y} \vee \neg z_{k+1})$ *préserve la cohérence d'arc généralisée de* $y \rightarrow \sum_{i=1}^{n} x_i \leqslant k$.

Démonstration. Dans le cas où y est affecté à $vrai$, la formule simplifiée représente la contrainte AtMostK encodée à l'aide des réseaux de tri. Maintenant, nous considérons deux cas en fonction de la valeur de vérité de z_{k+1}. Dans le premier cas, si z_{k+1} est affecté à $vrai$, alors nous déduisons $\neg y$ par propagation unitaire. En effet, comme les sorties sont triées par ordre décroissant, cela signifie que la contrainte AtMostK est falsifiée. Par conséquent, pour satisfaire l'AtMostK conditionnel, il faut assigner y à $faux$. Dans le second cas, si la valeur de vérité de z_{k+1} est $faux$, cela signifie que la contrainte AtMostK est vraie, par conséquent, peu importe la valeur de y, la contrainte AtMostK conditionnel est satisfaite. □

Encodage basé sur un compteur séquentiel unaire : Nous avons montré précédemment comment la contrainte AtMostOne conditionnelle peut être encodée en utilisant l'encodage basé sur un compteur séquentiel, tout en préservant la propriété GAC. Considérons maintenant le cas général de l'encodage dasé sue le compteur séquentiel de la contrainte AtMostK conditionnelle. Reprenant l'encodage basé sur le compteur séquentiel de la contrainte AtMostK présenté précédemment dans la section 1.2.3 :

$$(\neg x_1 \vee p_{1,1}) \tag{5.28}$$

$$\bigwedge_{1<j\leqslant k} \neg p_{1,j} \tag{5.29}$$

$$\bigwedge_{1<i<n} (\neg x_i \vee p_{i,1}) \wedge (\neg p_{i-1,1} \vee p_{i,1}) \tag{5.30}$$

$$\bigwedge_{1<i<n} \bigwedge_{1<j\leqslant k} (\neg x_i \vee \neg p_{i-1,j-1} \vee p_{i,j}) \wedge (\neg p_{i-1,j} \vee p_{i,j}) \tag{5.31}$$

$$\bigwedge_{1<i\leqslant n} (\neg x_i \vee \neg p_{i-1,k}) \tag{5.32}$$

Les clauses (5.28) ∧ (5.30) ∧ (5.31) nous permettent de propager toute assignation de x_i pour synchroniser tous les compteurs séquentiels intermédiaires, tandis que les clauses (5.29) ∧ (5.32) nous permettent de détecter toute incohérence de la contrainte AtMostK. En effet, en ajoutant $\neg y$ à toutes les clauses, les littéraux $p_{i,j}$ ne peuvent pas être propagés après n'importe quelle affectation de variables x_i ce qui empêche l'opération de synchronisation. Pour préserver la propriété GAC, nous devons ajouter $\neg y$ uniquement aux clauses de (5.29) ∧ (5.32) comme indiqué dans la formule suivante :

$$(\neg x_1 \vee p_{1,1}) \tag{5.33}$$

$$\bigwedge_{1<j\leqslant k} (\neg \mathbf{y} \vee \neg p_{1,j}) \tag{5.34}$$

$$\bigwedge_{1<i<n} (\neg x_i \vee p_{i,1}) \wedge (\neg p_{i-1,1} \vee s_{i,1}) \tag{5.35}$$

$$\bigwedge_{1<i<n} \bigwedge_{1<j\leqslant k} (\neg x_i \vee \neg p_{i-1,j-1} \vee p_{i,j}) \wedge (\neg p_{i-1,j} \vee p_{i,j}) \tag{5.36}$$

$$\bigwedge_{1<i\leqslant n} (\neg \mathbf{y} \vee \neg x_i \vee \neg p_{i-1,k}) \tag{5.37}$$

Exemple 5.12. *Considérons la contrainte suivante* $y \rightarrow x_1 + x_2 + x_3 \leqslant 2$ *encodée en CNF comme suit :* $(\neg x_1 \vee p_{1,1}) \wedge (\neg \mathbf{y} \vee \neg p_{1,2}) \wedge (\neg x_2 \vee p_{2,1}) \wedge (\neg p_{1,1} \vee p_{2,1}) \wedge (\neg x_2 \vee \neg p_{1,1} \vee p_{2,2}) \wedge (\neg p_{1,2} \vee p_{2,2}) \wedge (\neg \mathbf{y} \vee \neg x_2 \vee \neg p_{1,2}) \wedge (\neg \mathbf{y} \vee \neg x_3 \vee \neg p_{2,2})$
Supposons que nous commençons par l'affectation de x_1 *à* $vrai$ *alors, les littéraux* $p_{1,1}$ *et* $p_{2,1}$ *sont déduits par propagation unitaire. Ensuite, si nous assignons* x_2 *à* $vrai$*, le littéral* $p_{2,2}$ *est unitairement propagé. Finalement, en affectant* x_3 *à* $vrai$ *, ce qui viole la contrainte, le littéral* $\neg y$ *est propagé grâce à la dernière clause.*

Proposition 5.14. *L'encodage* (5.33) $\wedge$ (5.34) $\wedge$ (5.35) $\wedge$ (5.36) $\wedge$ (5.37) *préserve la cohérence d'arc généralisée de* $y \rightarrow \sum_{i=1}^{n} x_i \leqslant k$.

Démonstration. L'encodage basé sur un compteur séquentiel de la contrainte de cardinalité est également une formule de Horn. Par conséquent, nous pouvons appliquer le résultat de la proposition 5.10 pour conclure que $\neg y$ doit être ajouté uniquement aux clauses négatives afin de préserver la cohérence d'arc généralisée. La preuve est une simple généralisation de celle esquissée dans l'exemple 5.12. □

5.4 Expérimentations

5.4.1 Protocole expérimentale

Dans cette section, nous présentons une évaluation expérimentale comparative de l'approche proposée avec des algorithmes spécialisés dans l'extraction des règles d'association. Nous considérons les tâches de fouille de règles d'association suivantes :

1. Règles d'association classique « pure ».
2. Règles d'association fermés « closed ».
3. Règles d'association minimales non redondantes « MNR ».
4. Règles d'association indirectes.

Pour énumérer l'ensemble des modèles de la formule CNF résultante, nous suivons l'approche de [JLSS14]. L'algorithme d'énumération des modèles proposé est basé sur une procédure de recherche DPLL. Dans nos expérimentations, l'heuristique de choix de variable se concentre en priorité sur les variables X et Y respectivement pour sélectionner celle qui sera affectée par la suite. La puissance de cette approche consiste à utiliser la structure des littéraux observés pour effectuer efficacement la propagation unitaire. Notons également que les contraintes (5.5) et (5.7), dédiées au support et à la confiance, sont gérées dynamiquement sans transformation en forme CNF conduisant à un algorithme d'énumération de modèles hybrid SAT-CSP. En effet, les inéquations linéaires (5.5) et (5.7) sont gérées et propagées à la volée, comme se fait généralement en programmation par contraintes. Chaque modèle de la formule propositionnelle, encodant la tâche de fouille de règles d'association, correspond exactement

Données	#items	#trans	densité
Audiology	148	216	45%
Zoo-1	36	101	44%
Tic-tac-toe	27	958	33%
Anneal	93	812	45%
Australian-credit	125	653	41%
German-credit	112	1000	34%
Heart-cleveland	95	296	47%
Hepatitis	68	137	50
Hypothyroid	88	3247	49%
Kr-vs-kp	73	3196	49%
Lymph	68	148	40%
Mushroom	119	8124	18%
Primary-tumor	31	336	48%
Soybean	50	650	32%
Splice-1	287	3190	21%
Vote	48	435	33%

TABLE 5.3 – Caractéristiques des jeux de données.

à une règle d'association en considérant la valeur de vérité des variables propositionnelles qui encodent l'antécédent (X) et le conséquent (Y) de cette règle.

Afin d'évaluer la performance de notre approche SAT dédiée à la fouille des différentes variantes de règles d'association, Nous comparons notre outil à deux outils spécialisés dans l'extraction de règles d'association, à savoir *CORON*[5] et *SPMF*[6] [FVGG+14]. *CORON* et *SPMF* sont deux outils implémentés en Java qui intègrent une riche collection d'algorithmes de fouille de données. Pour les règles d'association *minimales non redondantes*, nous comparons notre approche à l'algorithme *ZART* implémenté dans *CORON* et *SPMF*. *ZART* est l'un des plus récents et efficaces algorithmes de l'état de l'art pour l'énumération des règles d'association minimales non redondantes [SNK06, Sza06]. Rappelons que *ZART* trouve les règles minimales non redondantes en deux étapes. Tout d'abord, l'ensemble de tous les itemsets fermés et fréquents ainsi que les générateurs minimaux sont extraits rapidement. Ensuite, l'identification des règles non redondantes est effectuée. Cette procédure à deux étapes consomme plus de temps. Les caractéristiques des différents jeux de données utilisés dans cette évaluation sont représentées dans la table 5.3, à savoir, le nombre d'items (#items), le nombre de transactions (#trans) et la densité.

Pour évaluer les performances de l'approche proposée, on varie le support et la confiance de 5% à 100% avec un intervalle de taille 5. Ainsi, un ensemble de 400 configurations est généré pour chaque jeu de données . Pour les règles d'association indirectes il y a un paramètre λ en plus. En effet, la fréquence, qui est égale au support divisé sur le nombre total de transaction, et la confiance sont variées de 20% à 100% avec un intervalle de 20%. λ est varié de 10% à 100% avec un intervalle de 10%. Ces variations génèrent 250 configurations pour chaque jeux de données.

Toutes les expérimentations ont été effectuées sur une machine Intel Xeon quad-core avec 32GB de RAM et une fréquence de 2.66 Ghz. On fixe une limite de temps de 15 minutes de temps CPU pour chaque instance (configuration).

5. Coron : http ://coron.loria.fr/site/system.php
6. SPMF : http ://www.philippe-fournier-viger.com/spmf/

Dans ces expérimentations, *SAT4Pure*, *SAT4Closed*, *SAT4MNR* et *SAT4Indirect* indiquent nos approches basées sur SAT pour la fouille des règles d'association classiques, fermées, minimales non redondantes et indirectes. De plus, nous considérons *SAT4MNR-D* qui partitionne la recherche comme dans [JSS15a]. Cela se fait comme suit : Soit $\Omega = \{a_1, \ldots, a_n\}$, on transforme le problème en n problème de fouille où chacun encode les règles $X \rightarrow Y$ tel que $\{a_1 \ldots, a_{i-1}\} \not\subset X$ et $a_i \in X$. On note aussi *SAT4MGR* notre outil basé sur SAT pour la fouille des règles les plus générales (Définition 5.2).

Pour donner une idée sur la taille de nos encodages pour l'extraction des règles d'association classiques, la plus petite (resp. la plus grande) formule correspond à l'encodage de zoo-1 (resp.mushroom) qui contient 274 variables et 4379 clauses (resp. 16486 variables et 1616795 clauses).

5.4.2 Résultats

Les Tables 5.4 et 5.5 décrivent nos résultats comparatifs. Pour chaque algorithme, on rapporte le nombre de configurations résolues ($\#S$), et le temps moyen de résolution ($avg.time$ en secondes). Pour chaque instance non résolue, le temps est mis à 900s (limite de temps). Dans la dernière ligne de la Table 5.5, on donne le nombre d'instances résolues et le temps CPU moyen global en secondes.

	SAT4Pure		CORON_Pure		SAT4Closed		CORON_Closed		SAT4Indirect		SPMF_Indirect	
Données	#S	avg. time(s)	#S	avg. time(s)	#S	avg. time(s)	#S	avg. time(s)	#S	avg. time(s)	#S	avg. time(s)
Audiology	20	855.00	20	855.01	20	855.00	20	855.01	124	453.74	61	680.45
Zoo-1	400	19.12	400	6.37	400	0.52	400	11.28	250	0.15	250	9.12
Tic-tac-toe	400	0.09	400	0.24	400	0.09	400	0.23	250	0.09	250	0.20
Anneal	101	709.50	101	678.41	147	604.09	103	679.31	171	309.69	55	702.04
Australian-credit	245	370.17	264	321.62	268	323.29	226	403.72	232	121.06	156	339.56
German-credit	306	246.88	322	192.52	329	198.02	304	238.79	244	49.07	210	154.49
Heart-cleveland	284	286.38	301	252.27	304	251.05	262	340.15	235	64.97	203	300.48
Hepatitis	305	241.41	304	228.00	324	206.02	266	312.26	245	32.98	205	187.92
Hypothyroid	85	732.12	121	665.41	107	686.95	64	761.59	163	336.40	81	621.29
Kr-vs-kp	172	552.92	203	487.73	192	523.66	146	590.89	204	206.47	114	499.33
Lymph	336	181.64	338	170.37	387	63.22	291	281.35	250	6.10	211	170.19
Mushroom	366	109.12	387	46.00	400	30.32	390	42.84	250	8.89	250	29.62
Primary-tumor	400	3.68	400	1.17	400	2.03	400	18.82	250	0.15	250	2.63
Soybean	400	2.90	400	1.50	400	0.17	400	7.94	250	0.05	250	0.76
Splice-1	380	53.44	400	3.52	380	54.04	400	3.25	250	61.73	250	0.50
Vote	380	66.74	400	1.46	400	32.40	398	30.22	250	0.84	250	1.48
Total	4560	279.76	**4741**	**247.29**	**4838**	**242.24**	4470	286.10	**3618**	**103.27**	3046	231.25

TABLE 5.4 – Règles d'association classiques, fermées et indirectes : `SAT` vs `CORON` et `SAT` vs `SPMF`

Règles d'association classiques : Les performances de *CORON* sont meilleures que *SAT4Pure*. Il résout 181 configurations de plus et il est meilleur sur tous les jeux de données considérés. Rappelons que *CORON* effectue l'énumération des règles d'association classiques en deux étapes. Nous avons observé que *CORON* effectue la première étape de manière efficace. Son temps CPU ne dépasse pas quelques secondes sur la majorité des configurations considérées. Pour les règles pures, la deuxième étape reste assez facile à effectuer. En effet, sur les règles d'association classiques, l'algorithme spécialisé *ZART* implémenté dans *CORON* est meilleur que SAT4Pure.

Les figures 5.1, 5.2, 5.3 et 5.4 illustrent le comportement des deux outils considérés sur deux jeux de données représentatifs, `Australian-credit` et`Kr-vs-kp`. Nous avons varié un paramètre, tout en maintenant l'autre fixe.

Pour les règles d'association classique, *CORON* et *SAT4Pure* présentent des comportements similaires. Lorsque la fréquence diminue, le temps nécessaire pour trouver toutes les règles augmente. Notons que pour certaines valeurs particulières de paramètres, notre approche peut surpasser *CORON* sur les règles classiques comme c'est le cas pour *Kr-vs-kp*.

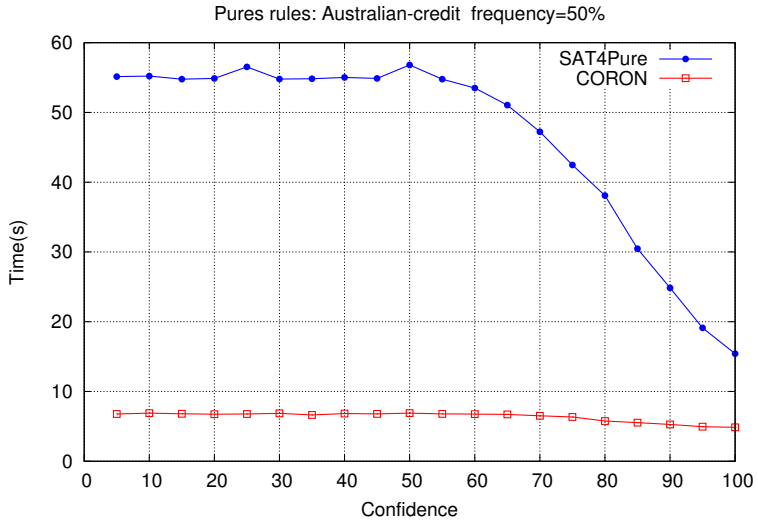


FIGURE 5.1 – Énumération des règles classiques : *Australian-credit* avec une fréquence de 50%

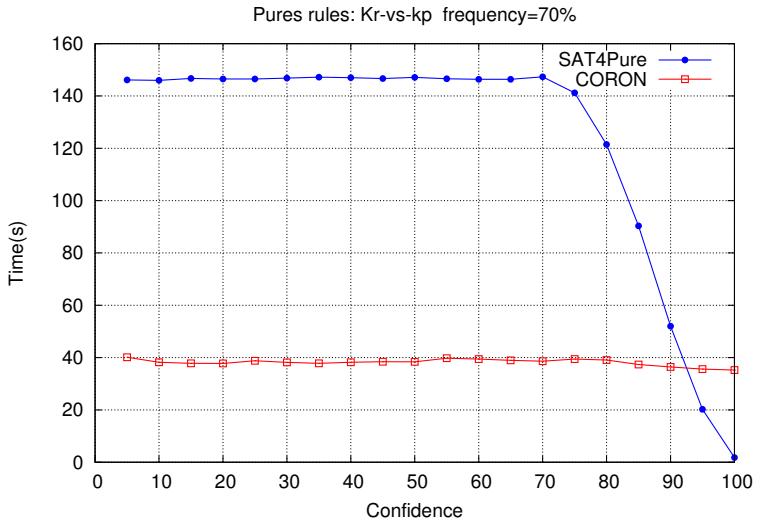


FIGURE 5.2 – Énumération des règles classiques : *Kr-vs-kp* avec une fréquence de 70%

Règles d'association fermées : Sur cette catégorie de règles, notre approche SAT est meilleur que *ZART*. Elle résout 368 configurations de plus que lui. A l'exception du jeux de données `Splice-1`, *SAT4Pure* est le meilleur sur toutes les données en termes de nombre de configurations résolues et de temps CPU moyen. Notons que pour le jeux de données Splice-1, le nombre de règles d'association fermées est très limité (inférieur à 4000). Ceci explique pourquoi *CORON* est meilleur que *SAT4Closed* sur ce jeu de données. Pour extraire les règles d'association fermées, *CORON* procède en deux étapes. Dans

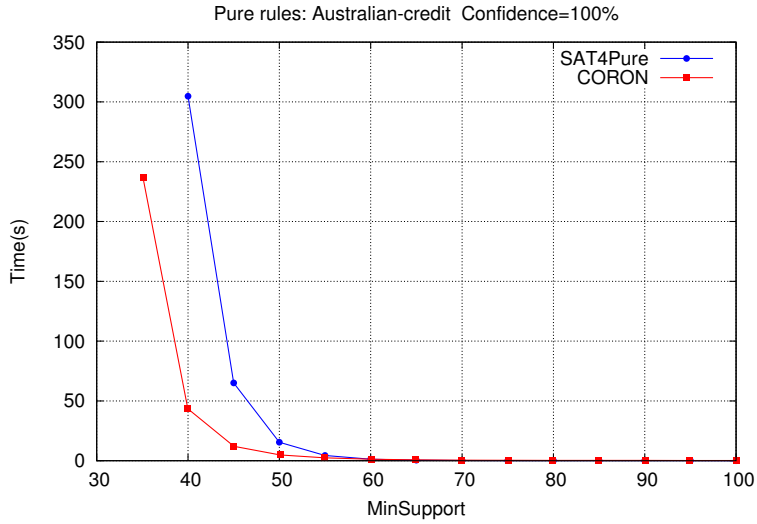


FIGURE 5.3 – Énumération des règles classiques : *Australian-credit* avec une confiance de 100%

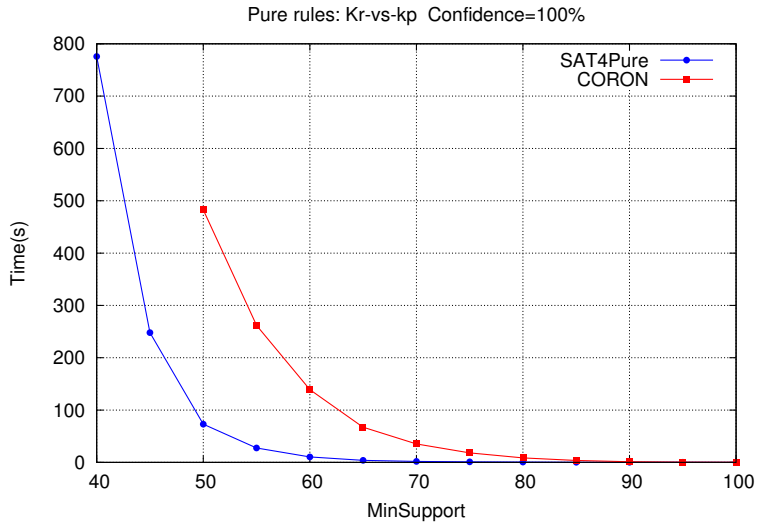


FIGURE 5.4 – Énumération des règles classiques : *Kr-vs-kp* avec une confiance de 100%

la première, l'ensemble de tous les itemsets fermés fréquents est efficacement énuméré (en quelques secondes), tandis que dans la deuxième étape, les règles d'association sont extraites à partir des itemsets fermés déjà trouvés. Cette dernière étape prend plus de temps. Par exemple, sur le jeu de données `Lymph`, l'approche SAT est remarquablement efficace. Elle résout environ 100 configurations de plus que *CORON*. Plus généralement, plus la densité des données est élevée, meilleures sont les performances de $SAT4Closed$.

Les figures 5.5, 5.6, 5.7 et 5.8 illustrent le comportement des deux outils considérés pour l'énumération des règles d'association fermées sur deux jeux de données représentatifs, `Australian-credit` et`Kr-vs-kp`. Lorsque la fréquence diminue, le temps nécessaire pour trouver toutes les règles augmente. L'approche basée sur SAT surpasser *CORON* sur les deux jeux de données `Kr-vs-kp` et `Australian-credit`. Cependant, nous pouvons noter que lorsque la confiance passe de 100% à 80%,

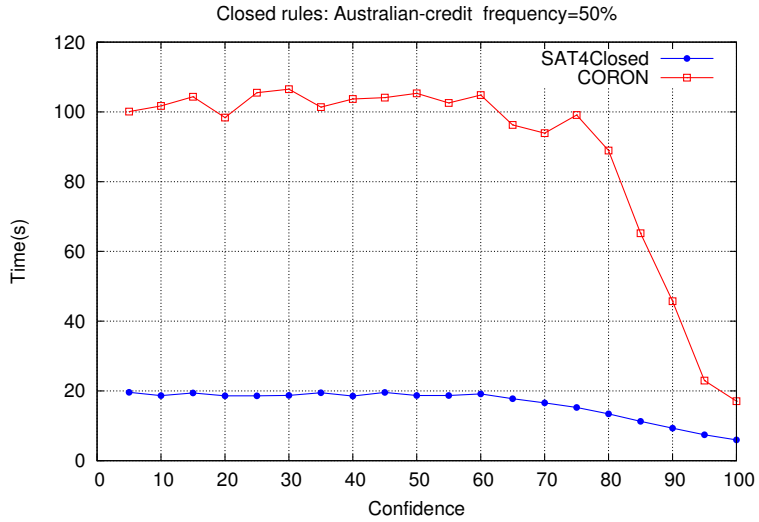


FIGURE 5.5 – Énumération des règles fermées : *Australian-credit* avec une fréquence de 50%

le temps CPU augmente considérablement. Un tel écart est plus visible avec *SAT4Closed* sur les règles d'association classiques et avec *CORON* sur les règles d'association fermées. En effet, pour l'instance `Kr-vs-kp`, en utilisant *SAT4Closed*, la variation est entre 0 et 70 secondes tandis qu'avec *CORON*, la temps CPU va de 40 à 240 secondes.

Règles d'association indirectes : Pour les règles indirectes, les performances de l'approche basée sur SAT sont très impressionnantes. L'outil *SAT4Indirect* résout 572 instances de plus que *SPMF*. Là encore, l'approche est meilleure sur tous les jeux de données considérés. Comme on peut le remarquer, le temps nécessaire à *SAT4Indirect* pour obtenir toutes les règles d'association indirectes est relativement stable et très faible par rapport à *SPMF*. Le nombre de d'associations indirectes est très faible par rapport aux règles d'association classiques ou fermées. Cependant, *SPMF* prend beaucoup de temps pour les trouver. Par exemple, si nous prenons le jeux de données `Hepatitis` avec les seuils : $frquence = 40\%$, $confiance = 40\%$ et $\lambda = 20\%$, *SPMF* prend $122,56$ secondes pour trouver seulement 359 règles d'association indirectes, alors que *SAT4Indirect* ne dépasse pas 1 seconde.
Nous avons également remarqué que pour certaines configurations, *SPMF* prend trop de temps CPU sans trouver de règle d'association indirecte sous la limite de temps de 900 secondes. En résumé, sur les règles d'association indirecte, *SAT4Indirect* surpasse *SPMF*.

Règles d'association non redondantes : Selon les résultats de la Table 5.5, *SAT4MNR* surpasse les deux outils spécialisés *CORON* et *SPMF*. Il résout 488 configurations de plus que *CORON* et 920 de plus que *SPMF*. *SAT4MNR-D* est le meilleur sur tous les jeux de données en termes de nombre de configurations résolues et en temps CPU moyen, à l'exception de *mushroom* où *CORON* est meilleur en temps mais *SAT4MNR-D* résout toutes les configuration. Remarquons que pour *mushroom*, le nombre de règles d'association minimales non redondantes est très limité. Cela explique pourquoi *SAT4MNR* n'est pas meilleur que *CORON* sur ce jeu de données. Par exemple, pour le jeu de données *anneal*, *SAT4MNR* est remarquablement efficace. Il résout environ 100 configurations de plus que *CORON* et environ 200 configurations de plus que *SPMF*. On peut aussi remarquer que pour le jeu de données *Lymph*, *SAT4MNR-D* résout toutes les configurations en un temps moyen de $7s$ tandis que *CORON* et *SPMF* ne peuvent pas résoudre toute les configurations et ils prennent beaucoup de temps comparés à *SAT4MNR-D*. Plus gé-

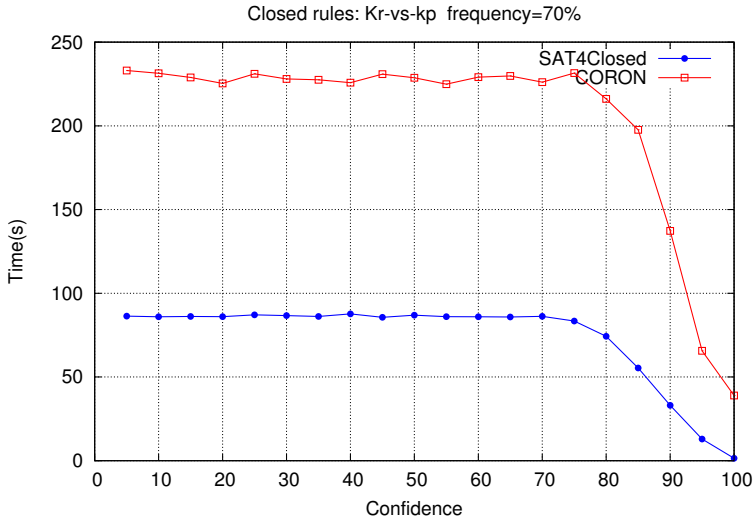


FIGURE 5.6 – Énumération des règles fermées : *Kr-vs-kp* avec une fréquence de 70%

	SAT4MNR-D		*SAT4MNR*		*CORON*		*SPMF*		*SAT4MGR*	
Données	#S	avg. time(s)	#S	avg. time(s)	#S	avg. time(s)	#S	avg. time(s)	#S	avg. time(s)
Audiology	21	854,82	21	854.87	20	855.01	20	855.00	20	855.00
Zoo-1	400	0.23	400	0.27	400	1.35	373	108.60	400	0.71
Tic-tac-toe	400	0.34	400	0.14	400	0.24	400	0.20	400	0.61
Anneal	279	337.25	248	405.82	160	591.39	80	724.46	221	461.05
Australian-credit	298	265.74	278	309.32	251	352.01	220	417.94	263	358.40
German-credit	354	149.03	328	212.58	321	206.34	278	294.45	304	272.88
Heart-cleveland	331	200.28	317	235.79	271	307.57	240	368.21	286	289.28
Hepatitis	360	140.69	343	170.89	286	284.09	260	331.57	315	228.13
Hypothyroid	150	615.13	126	649.22	104	681.52	80	751.23	109	676.03
kr-vs-kp	198	504.62	172	556.85	168	552.04	140	627.64	158	583.25
Lymph	400	6.78	400	19.21	357	131.07	280	316.78	395	37.15
Mushroom	400	146.87	389	77.02	400	3.81	360	97.25	354	181.89
Primary-tumor	400	2.08	400	4.61	400	4.15	379	87.66	400	8.11
Soybean	400	0.36	400	0.20	400	0.61	380	48.51	400	2.26
Vote	400	5.43	400	30.46	364	87.56	380	84.82	372	111.06
Total	**4790**	**215.31**	**4622**	**235.15**	4302	270.58	3870	340.94	4397	271.05

TABLE 5.5 – Règles d'association non redondantes : *SAT4MNR* vs *CORON* vs *SPMF*

néralement, plus la densité des données est élevée, meilleures sont les performances de *SAT4MNR*. Il est intéressant de noter que le partitionnement du problème de fouille permet de pousser davantage les performances de *SAT4MNR*. En effet, *SAT4MNR-D* permet d'obtenir de meilleures performances, c-à-d, 168 instances résolues et le temps de résolution moyen est amélioré de 235.15 à 215.31. *SAT4MGR* résout moins de configurations que *SAT4MNR* car l'ensemble de règles minimales non redondantes est connu pour être plus réduit comparé à celui des règles les plus générales.

Les figures 5.9, 5.10, 5.11 et 5.12 illustrent le comportement des approches considérées pour l'extraction de règles d'association minimales non redondantes sur deux jeux de données représentatifs `Anneal` et `Kr-vs-kp`. Les résultats sont obtenus en variant un paramètre, tout en maintenant les autres fixés. Lorsque le seuil minimum de support diminue, le temps nécessaire pour trouver toutes les règles augmente. Remarquons que pour *CORON* et *SPMF* le temps augmente rapidement comparé à *SAT4MNR-D*. Pour *anneal*, *SPMF* (resp. *CORON*) n'est pas capable d'extraire toutes les règles non redondantes lorsque

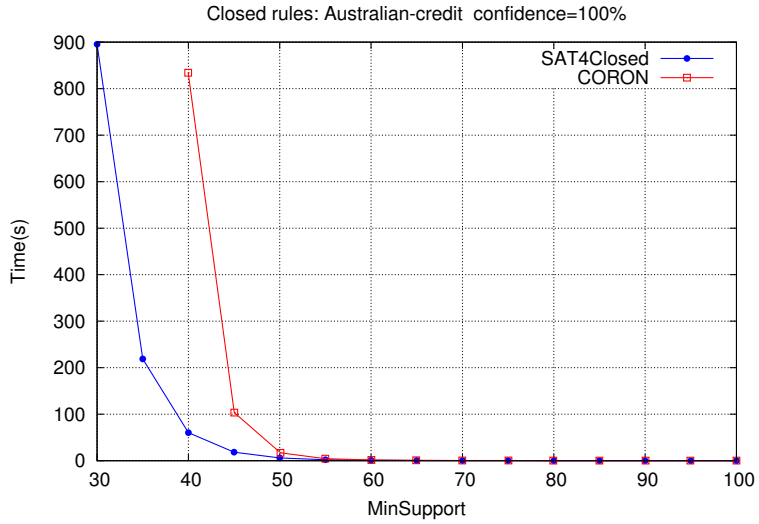


FIGURE 5.7 – Énumération des règles fermées : *Australian-credit* avec une confiance de 100%

le seuil minimum de support est inférieur à 85%(resp. 65%). Par contre, avec *SAT4MNR* et *SAT4MNR-D* il est possible d'obtenir toutes les règles pour toutes les valeurs du seuil minimum de support. Pour `Kr-vs-kp`, il est important de noter que le temps nécessaire pour extraire les règles augmente considérablement pour *SPMF* et *CORON* même pour une confiance élevée. Par exemple, quand le support minimum va de 100% à 80% le temps est multiplié au moins par un facteur de 10. Une telle augmentation est très limitée pour *SAT4MNR* et *SAT4MNR-D*.

Enfin, la Table 5.6 présente la variation du rapport entre le nombre de règles classiques (pures), les règles fermées, les règles non redondantes les plus générales et les règles minimales non redondantes pour le jeu de données `Kr-vs-kp`. Comme on peut l'observer, le nombre de règles minimales non redondantes est plus petit que celui des règles non redondantes les plus générales. Ce dernier est plus petit que le nombre de règles d'association fermées, qui est en soi plus petit que celui des pures. Par exemple, lorsque le seuil minimum de support est égal à 40%, les règles d'association minimales non redondantes présentent 2.85% de toutes les règles d'association classiques tandis que les règles non redondantes les plus générales représentent environ 3.90%.

Seuil minimum de support (%)	40	45	50	55	60	65	70
$\#Classiques/\#Fermes$	7.67	5.68	3.64	2.99	2.46	1.95	1.67
$\#Fermes/\#MGRs$	2.40	2.16	1.95	1.78	1.61	1.46	1.35
$\#MGRs/\#MNRs$	1.94	1.83	1.73	1.63	1.54	1.45	1.38

TABLE 5.6 – `Kr-vs-kp` : Règles d'association classiques vs fermées vs MNRs vs MGRs

5.5 Conclusion

Dans ce chapitre, nous avons développé une nouvelle approche de fouille de règles d'association qui permet d'extraire efficacement les règles d'association. Notre approche déclarative est différente de toutes les techniques existantes car l'extraction des règles d'association est réalisée en une seule étape grâce à notre encodage basé sur SAT. Comme deuxième contributions, nous avons montré que notre

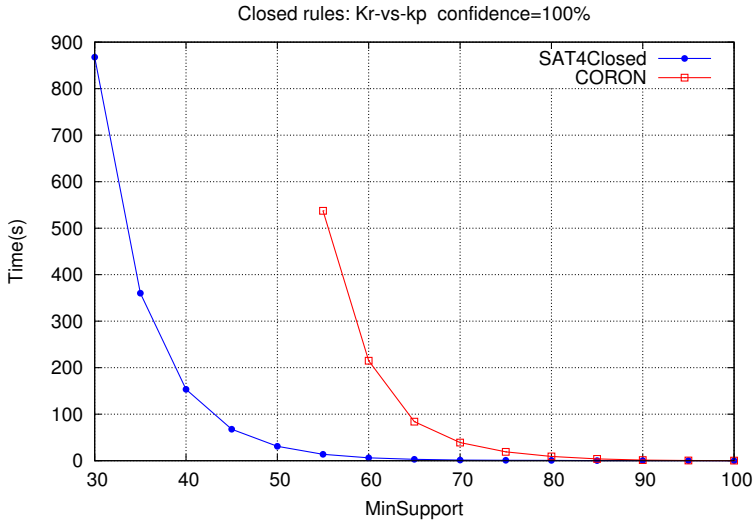


FIGURE 5.8 – Énumération des règles fermées : *Kr-vs-kp* avec une confiance de 100%

cadre proposé est flexible et déclaratif en modélisant facilement d'autres variantes importantes, telles que l'extraction de règles d'association fermées, indirectes, minimales non redondantes et les plus générales. Ensuite, pour accélérer l'énumération des règles non redondantes, nous avons proposés des encodages efficaces pour la contrainte de cardinalité conditionnelle. Enfin, des expérimentations sur de nombreux jeux de données montrent que notre approche permet d'obtenir de meilleures performances par rapport aux approches spécialisées sur la majorité des variantes de règles d'association.

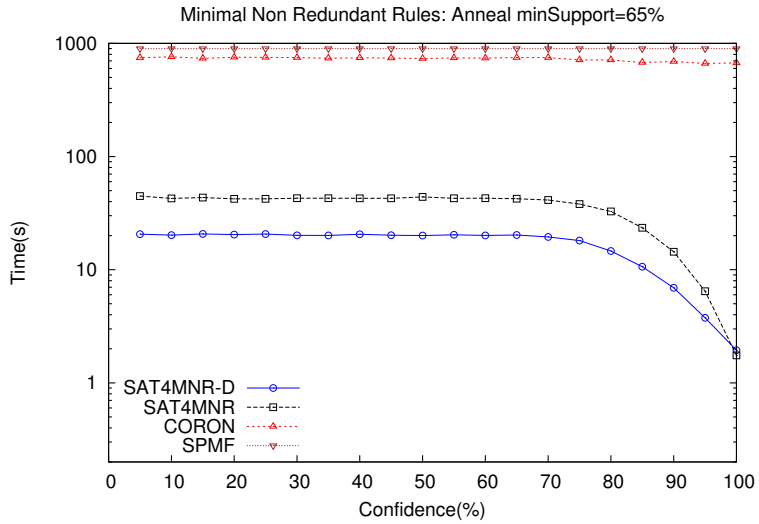


FIGURE 5.9 – Énumération des règles MNR : *Anneal* avec une fréquence de 65%

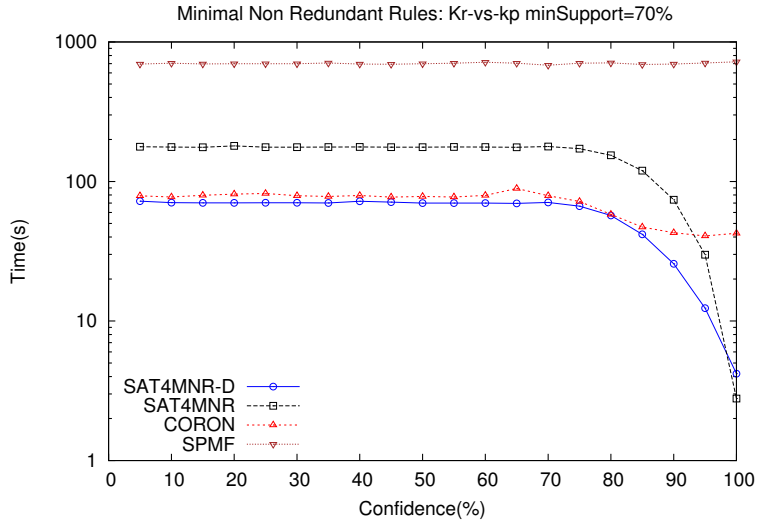


FIGURE 5.10 – Énumération des règles MNR : *Kr-vs-kp* avec une fréquence de 70%

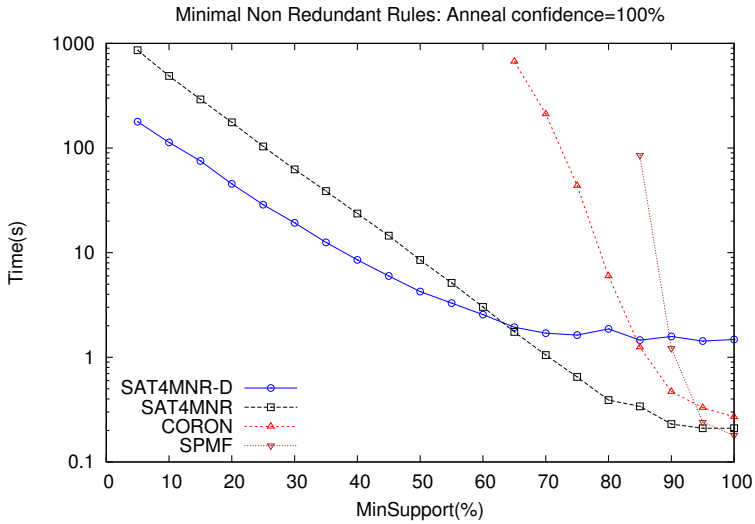


FIGURE 5.11 – Énumération des règles MNR : *Anneal* avec une confiance de 100%

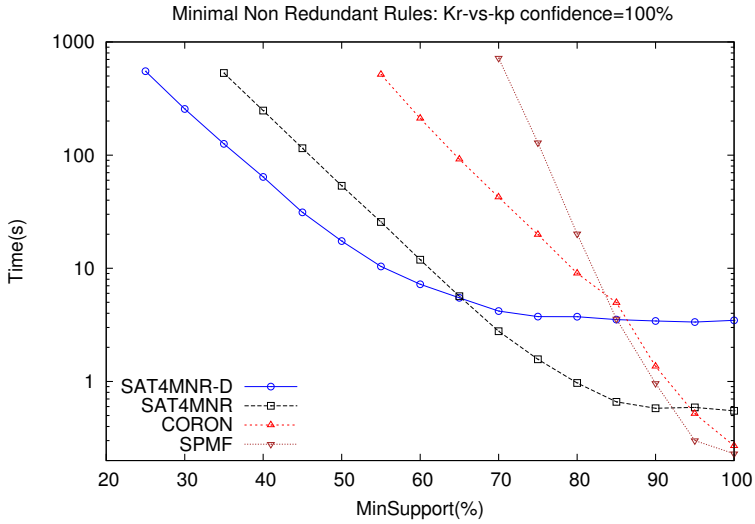


FIGURE 5.12 – Énumération des règles MNR : *Kr-vs-kp* avec une confiance de 100%

Conclusion générale

Dans cette thèse nous nous sommes intéressés à la fertilisation croisée entre l'intelligence artificiel et la fouille de données. Nous avons considéré deux problèmes fondamentaux en fouille de données, à savoir, le clustering et l'extraction de règles d'association. Chacun des deux problèmes occupe une place très importante dans plusieurs domaines d'application. Nos contributions pour le clustering consistent principalement à montrer d'abord que la logique propositionnelle peut être utilisée pour représenter des données complexes et proposer par la suite des algorithmes dédiés a ce nouveau type de données. Tandis que nos contributions dans le domaine de fouille de règles d'association consistent à proposer un cadre déclaratif basé sur la satisfiablité propositionnelle pour l'extraction de différentes variantes de règles d'association.

Dans le cadre de clustering des formules propositionnelles, nous avons commencé par donner les contraintes qu'une solution de ce problème doit satisfaire comme la cohérence de chaque cluster. Puis, en utilisant la représentation sémantique des formules via leurs ensembles de modèles, nous avons adapté les deux algorithmes : $k\text{-}means$ et l'algorithme hiérarchique ascendant. Nous avons montré que la mesure de similarité de Jaccard peut être utilisée pour mesurer la proximité entre deux formules. Pour préserver la cohérence dans chaque cluster, nous avons proposé la conjonction des formules appartenant au cluster comme représentant. En effet, les deux algorithmes ont été adapté. Cependant, les deux algorithmes ne garantissent pas l'obtention d'une solution. Pour cette raison, nous avons proposé un algorithme hiérarchique descendant qui permet de trouver un clustering cohérent. Cet algorithme se base principalement sur le calcul d'un transversal minimum pour diviser un cluster donné. L'étape de division est effectuée sur le mauvais cluster jusqu'à satisfaction d'un critère d'arrêt. Enfin, nous avons proposé un encodage qui nous permet d'effectuer la division sur les formules propositionnelle sans avoir besoin d'énumérer leurs modèles. Cet encodage réduit le nombre d'appels à un oracle de comptage et remplace le calcul d'un transversal minimum par des appel à un oracle SAT. Des expérimentations préliminaires montrent la faisabilité des approches proposées.

Dans le cadre de fouille de règles d'association, nous avons commencé par encoder le problème d'extraction de règles d'association standard en utilisant la satisfiabilité propositionnelle. Nous avons montré par la suite comment le problème de fouille de règles fermées qui est une représentation condensée des règles d'association, peut être facilement encodé en rajoutant au premier encodage celui de la contrainte de fermeture. Ensuite, pour montrer la flexibilité de l'approche, nous avons considérer d'autres variantes de règles d'association, à savoir, les règles indirectes, les règles minimales non redondantes et les règles les plus générales. Premièrement, nous avons montré comment l'encodage proposé pour les règles standard peut être adapté pour extraire les règles indirectes. Deuxièmement, pour encoder les règles minimales non redondantes, nous avons commencé par proposer le premier encodage pour extraire les itemsets minimaux (libres). Nous avons montré aussi, comment améliorer cet encodage en utilisant des contraintes AtmostOne conditionnelles. Puis, en montrant que les règles minimales non redondantes correspondent au règles où les antécédents sont des itemsets minimaux et l'union des antécédents et des conséquences sont des itemsets fermés, nous avons utilisé les encodages déjà proposés pour ces deux contraintes pour capturer cet ensemble de règles. De la même façon, nous avons montré comment utiliser

l'encodage des itemsets minimaux pour extraire les règles les plus générales en utilisant SAT. Enfin, nous avons proposé des adaptations des encodages existants pour la contrainte de cardinalité pour encoder la contrainte de cardinalité conditionnelle. Nous avons utilisé ces adaptations pour énumérer efficacement les règles minimales non redondantes. Les expérimentations sur un large ensemble de données ont montré que sur les bases de données denses, l'approche proposée est largement meilleure que les algorithmes dédiés et plus particulièrement pour le cas des règles minimales non redondantes.

Perspectives : Ce travail ouvre de nombreuses perspectives pour de futurs travaux de recherche :

— *Clustering des formules propositionnelles :* Nous avons montré dans ce travail que la logique propositionnelle peut être un bon moyen de représentation des données complexes. Elle offre un langage qui permet l'expression de plusieurs types de contraintes. Puis, nous avons défini le problème de clustering sur ce type de données et nous avons adapté une mesure de proximité basée sur le comptage de modèles que nous avons utilisé pour adapter des algorithmes existants. Cependant, la mesure adaptée est très difficile à calculer. En effet, la première perspective consiste à adapter ou définir d'autres mesures de similarité facilement calculables pour le clustering des formules propositionnelles comme la distance de Hamming. Ensuite, nous avons proposé un algorithme hiérarchique ascendant où le cluster avec le moins de modèles partagés est divisé. Cette mesure de qualité de clusters est très sensible aux formules qui sont des bruits et elle ne considère pas la structure globale du cluster. En effet, trouver un bon critère de division qui prend en compte les structures des clusters est une question très importante dans ce travail. De plus, permettre les chevauchements entre les clusters jusqu'à la fin peut influencer la qualité des clusters et par conséquent, le comportement de l'algorithme. En effet, éliminer les chevauchement après chaque étape de division ou les exploiter efficacement pour améliorer la qualité du clustering fourni est une piste à creuser. Enfin, dans ce cadre nous avons considéré que toutes les contraintes doivent être satisfaites. Alors que dans certaines applications les agents peuvent exprimer des préférences entre les modèles en précisant des préférences sur les contraintes. Par exemple, on peut permettre les expressions suivantes : si la voiture est rouge, alors je serai plus satisfait que si elle est bleue. Ceci permet de définir un ordre sur l'ensemble des modèles. Nous pouvons considérer ce cadre en permettant de rajouter des poids sur les contraintes comme c'est le cas dans le problème Max-SAT pondéré. Développer un cadre de clustering pour des formules où les clauses ont des poids est très intéressant.

— *Satisfiabilité pour la fouille des règles d'association :* Nous avons proposé des encodages basés sur la satisfiabilité pour différentes variantes de problèmes de fouille de règles d'association. Cependant, nous avons remarqué que les tailles des formules CNF générées, augmentent rapidement avec l'augmentation du nombre de transactions ou du nombre d'items. En effet, encoder ces problèmes en SAT pour de grandes bases de données transactionnelles n'est pas faisable. Pour résoudre ce problème nous envisageons d'explorer les deux pistes suivantes. Premièrement, nous avons remarqué que dans les encodages des contraintes de couverture et de minimalité, il y a des sous-ensembles de variables qui se répètent souvent ce qui génère des sous formules redondantes. En effet, nous envisageons d'abord de détecter ces sous formules redondantes et rajouter des variables propositionnelles équivalentes à chacune, puis toutes les apparitions des sous formules sont remplacées par leurs variables correspondantes. Cette méthode de compression permet de réduire largement la formule CNF encodant le problème de fouille de règles d'association. Par conséquent, la résolution de cette formule est accélérée car la propagation unitaire va l'être aussi. Deuxièmement, nous avons constaté que la décomposition de la base de données en sous bases permet d'avoir des petites formules même pour de grands jeux de données et surtout si les données ne sont pas denses. Pour cette raison, nous envisageons d'explorer l'idée de développer

des techniques de décomposition et de parallélisation des encodages pour accélérer encore plus l'énumération des règles d'association. Enfin, proposer des encodages SAT pour le problème de fouille de règles d'association sur d'autres types de données comme les séquences constitue une piste intéressante.

Table des figures

Liste des tableaux

Bibliographie

[AGF+09] Gowtham ATLURI, Rohit GUPTA, Gang FANG, Gaurav PANDEY, Michael STEINBACH, et Vipin KUMAR. Association analysis techniques for bioinformatics problems. Dans *Bioinformatics and Computational Biology*, pages 1–13. Springer, 2009. (document), 3

[Agg15] Charu C AGGARWAL. *Data mining : the textbook*. Springer, 2015. 2.1.1, 2.1.1, 2.1.3, 2.1.3, 2.3, 7, 3.1.1, 12

[AGGR98] Rakesh AGRAWAL, Johannes GEHRKE, Dimitrios GUNOPULOS, et Prabhakar RAGHAVAN. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998. 5

[AGS16] John O. R. AOGA, Tias GUNS, et Pierre SCHAUS. « An Efficient Algorithm for Mining Frequent Sequence with Constraint Programming ». Dans *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II*, pages 315–330, 2016. 3.3

[ALMS11] Gilles AUDEMARD, Jean-Marie LAGNIEZ, Bertrand MAZURE, et Lakhdar SAÏS. « On freezing and reactivating learnt clauses ». Dans *International Conference on Theory and Applications of Satisfiability Testing*, pages 188–200. Springer, 2011. 4

[ANORC09] Roberto ASÍN, Robert NIEUWENHUIS, Albert OLIVERAS, et Enric RODRÍGUEZ-CARBONELL. « Cardinality networks and their applications ». Dans *International Conference on Theory and Applications of Satisfiability Testing*, pages 167–180. Springer, 2009. 1.2.3

[ANORC11] Roberto ASÍN, Robert NIEUWENHUIS, Albert OLIVERAS, et Enric RODRÍGUEZ-CARBONELL. « Cardinality networks : a theoretical and empirical study ». *Constraints*, 16(2) :195–221, 2011. 1.2.3, 5.1

[AR13] Charu C AGGARWAL et Chandan K REDDY. *Data clustering : algorithms and applications*. CRC press, 2013. 2.1.5, 6, 8, 11

[AS94] Rakesh AGRAWAL et Ramakrishnan SRIKANT. « Fast Algorithms for Mining Association Rules in Large Databases ». Dans *Proceedings of VLDB'94*, pages 487–499, 1994. (document), 3, 3.1.1, 12, 3.2, 5

[AS09] Gilles AUDEMARD et Laurent SIMON. « Predicting Learnt Clauses Quality in Modern SAT Solvers. ». Dans *IJCAI*, volume 9, pages 399–404, 2009. 1

[ASBP13] Ehsan ABBASNEJAD, Scott SANNER, Edwin V BONILLA, et Pascal POUPART. « Learning Community-Based Preferences via Dirichlet Process Mixtures of Gaussian Processes. ». Dans *IJCAI*, pages 1213–1219, 2013. 2

[AV07] David ARTHUR et Sergei VASSILVITSKII. « k-means++ : The advantages of careful seeding ». Dans *Proceedings of the eighteenth annual ACM-SIAM symposium on Dis-*

crete algorithms, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. 8

[Bac02] Fahiem BACCHUS. « Enhancing Davis Putnam with Extended Binary Clause Reasoning ». Dans *Eighteenth National Conference on Artificial Intelligence*, pages 613–619, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. 1.2.1

[BB92] Michael BURO et H Kleine BÜNING. *Report on a SAT competition*. Fachbereich Math.-Informatik, Univ. Gesamthochschule, 1992. 11

[BB03] Olivier BAILLEUX et Yacine BOUFKHAD. « Efficient CNF encoding of boolean cardinality constraints ». Dans *International conference on principles and practice of constraint programming*, pages 108–122. Springer, 2003. 1.2.3

[BBR00] Jean-Francois BOULICAUT, Artur BYKOWSKI, et Christophe RIGOTTI. « Approximation of frequency queries by means of free-sets ». Dans *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 75–85. Springer, 2000. 3.1.2, 3.1.2

[BBR09] Olivier BAILLEUX, Yacine BOUFKHAD, et Olivier ROUSSEL. « New encodings of pseudo-boolean constraints into CNF ». Dans *International Conference on Theory and Applications of Satisfiability Testing*, pages 181–194. Springer, 2009. 1.2.3, 5.1

[BCG01] Douglas BURDICK, Manuel CALIMLIM, et Johannes GEHRKE. « Mafia : A maximal frequent itemset algorithm for transactional databases ». Dans *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 443–452. IEEE, 2001. 3.1.2

[BCMG04] Fernando BERZAL, Juan-Carlos CUBERO, Nicolas MARIN, et Matias GAMEZ. « Anomalous association rules ». Dans *IEEE ICDM workshop alternative techniques for data mining and knowledge discovery*, 2004. 13

[BDW08] Sugato BASU, Ian DAVIDSON, et Kiri WAGSTAFF. *Constrained clustering : Advances in algorithms, theory, and applications*. CRC Press, 2008. 13

[BG06] Arindam BANERJEE et Joydeep GHOSH. « Scalable clustering algorithms with balancing constraints ». *Data Mining and Knowledge Discovery*, 13(3) :365–395, 2006. 2.4

[BGM+97] Yacine BOUFKHAD, Éric GRÉGOIRE, Pierre MARQUIS, Bertrand MAZURE, et Lakhdar SAÏS. « Tractable Cover Compilations ». Dans *Proceedings of the Fifteenth International Joint Conference on Artificial Intell igence (IJCAI'97)*, volume 1, pages 122–127, Nagoya (Japan), août 1997.

[BGS99] Laure BRISOUX, Éric GRÉGOIRE, et Lakhdar SAIS. « Improving backtrack search for SAT by means of redundancy ». Dans *International Symposium on Methodologies for Intelligent Systems*, pages 301–309. Springer, 1999. 2

[BHS94] Endre BOROS, Peter L HAMMER, et Xiaorong SUN. « Recognition of q-Horn formulae in linear time ». *Discrete Applied Mathematics*, 55(1) :1–13, 1994. 11

[BJ98] Roberto J BAYARDO JR. « Efficiently mining long patterns from databases ». *ACM Sigmod Record*, 27(2) :85–93, 1998. 3.1.2, 3.1.2

[BJS97] Roberto J BAYARDO JR et Robert SCHRAG. « Using CSP look-back techniques to solve real-world SAT instances ». Dans *Aaai/iaai*, pages 203–208, 1997. 1.2.2, 11

[BKS04] Paul BEAME, Henry KAUTZ, et Ashish SABHARWAL. « Towards understanding and harnessing the potential of clause learning ». *Journal of Artificial Intelligence Research*, 22 :319–351, 2004. 11

[BMvL12] Sébastien BUBECK, Marina MEILĂ, et Ulrike von LUXBURG. « How the initialization affects the stability of the k-means algorithm ». *ESAIM : Probability and Statistics*, 16 :436–452, 2012. 4.2.1

[BPT+00] Yves BASTIDE, Nicolas PASQUIER, Rafik TAOUIL, Gerd STUMME, et Lotfi LAKHAL. « Mining Minimal Non-redundant Association Rules Using Frequent Closed Itemsets ». Dans *Computational Logic - CL 2000*, volume 1861, pages 972–986, 2000. (document), 3.1.2, 3.1.2, 3.1.2, 3.1.2, 3.2.2, 3.2.2, 5, 5.2, 5.2.3, 5.2.3

[Bru80] Maurice BRUYNOOGHE. « Analysis of dependencies to improve the behaviour of logic programs ». Dans *International Conference on Automated Deduction*, pages 293–305. Springer, 1980. 11

[BW03] Fahiem BACCHUS et Jonathan WINTER. « Effective preprocessing with hyper-resolution and equality reduction ». Dans *International conference on theory and applications of satisfiability testing*, pages 341–355. Springer, 2003. 11

[CDS96] Marco CADOLI, Francesco M. DONINI, et M. SCHAERF. « Is intractability of non-monotonic reasoning a real drawback ? ». *Artificial Intelligence*, 88(1-2) :215–256, 1996.

[CH74] Tadeusz CALIŃSKI et Jerzy HARABASZ. « A dendrite method for cluster analysis ». *Communications in Statistics-theory and Methods*, 3(1) :1–27, 1974. 8

[CJSS12] Emmanuel COQUERY, Said JABBOUR, Lakhdar SAIS, et Yakoub SALHI. « A SAT-Based Approach for Discovering Frequent, Closed and Maximal Patterns in a Sequence. ». Dans *ECAI*, volume 242, pages 258–263, 2012. 3.3

[CKV13] M Emre CELEBI, Hassan A KINGRAVI, et Patricio A VELA. « A comparative study of efficient initialization methods for the k-means clustering algorithm ». *Expert systems with applications*, 40(1) :200–210, 2013. 8

[CM88a] K. M. CHANDY et J. MISRA. *Parallel program design : a foundation*. Addison-Wesley Publishing Company, 1988.

[CM88b] K. M. CHANDY et J. MISRA. *Parallel program design : a foundation*. Addison-Wesley Publishing Company, 1988.

[CMV13] Supratik CHAKRABORTY, Kuldeep S MEEL, et Moshe Y VARDI. « A scalable approximate model counter ». Dans *International Conference on Principles and Practice of Constraint Programming*, pages 200–216. Springer, 2013. 4.2.1

[Coo71] Stephen A. COOK. « The Complexity of Theorem-proving Procedures ». Dans *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM. 1, 1.2

[CRB06] Toon CALDERS, Christophe RIGOTTI, et Jean-François BOULICAUT. A survey on condensed representations for frequent sets. Dans *Constraint-based mining and inductive databases*, pages 64–80. Springer, 2006. 3.1.2

[DABC93] Olivier DUBOIS, Pascal ANDRÉ, Yacine BOUFKHAD, et Jacques CARLIER. « SAT versus UNSAT ». Dans *Cliques, Coloring, and Satisfiability, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, October 11-13, 1993*, pages 415–436, 1993. 11, 11

[DBB08] Ayhan DEMIRIZ, Kristin P BENNETT, et Paul S BRADLEY. « Using assignment constraints to avoid empty clusters in k-means clustering ». *Constrained clustering : advances in algorithms, theory, and applications*, page 201, 2008. 2.4

[DC03] Nicolas DURAND et Bruno CREMILLEUX. Ecclat : a new approach of clusters discovery in categorical data. Dans *Research and Development in Intelligent Systems XIX*, pages 177–190. Springer, 2003. 3.1.2

[DD06] Gilles DEQUEN et Olivier DUBOIS. « An efficient approach to solving random k-SAT problems ». *Journal of Automated Reasoning*, 37(4) :261–276, 2006. 11

[DDV13] Thi-Bich-Hanh DAO, Khanh-Chuong DUONG, et Christel VRAIN. « A declarative framework for constrained clustering ». Dans *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 419–434. Springer, 2013. (document), 2.4.2

[DDV15] Thi-Bich-Hanh DAO, Khanh-Chuong DUONG, et Christel VRAIN. « Constrained minimum sum of squares clustering by constraint programming ». Dans *International Conference on Principles and Practice of Constraint Programming*, pages 557–573. Springer, 2015. 2.4.2

[DDV17] Thi-Bich-Hanh DAO, Khanh-Chuong DUONG, et Christel VRAIN. « Constrained clustering by constraint programming ». *Artif. Intell.*, 244 :70–94, 2017. 2.4.2

[Dec90] Rina DECHTER. « Enhancement schemes for constraint processing : Backjumping, learning, and cutset decomposition ». *Artificial Intelligence*, 41(3) :273–312, 1990. 11

[DH80] Michel DELATTRE et Pierre HANSEN. « Bicriterion cluster analysis ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (4) :277–291, 1980. 7

[Dic45] Lee R DICE. « Measures of the amount of ecologic association between species ». *Ecology*, 26(3) :297–302, 1945. 2.1.2

[DLL62] Martin DAVIS, George LOGEMANN, et Donald LOVELAND. « A machine program for theorem-proving ». *Communications of the ACM*, 5(7) :394–397, 1962. 1.2.2, 9, 11, 11

[DP60] Martin DAVIS et Hilary PUTNAM. « A computing procedure for quantification theory ». *Journal of the ACM (JACM)*, 7(3) :201–215, 1960. 1.2.2

[DR05] Ian DAVIDSON et SS RAVI. « Clustering with constraints : Feasibility issues and the k-means algorithm ». Dans *Proceedings of the 2005 SIAM international conference on data mining*, pages 138–149. SIAM, 2005. 2.4, 13

[DR07] Ian DAVIDSON et SS RAVI. « The complexity of non-hierarchical clustering with instance and cluster level constraints ». *Data mining and knowledge discovery*, 14(1) :25–61, 2007. 2.4

[DRS10a] Ian DAVIDSON, S. S. RAVI, et Leonid SHAMIS. « A SAT-based Framework for Efficient Constrained Clustering ». Dans *Proceedings of SDM'10*, pages 94–105, 2010. (document)

[DRS10b] Ian DAVIDSON, SS RAVI, et Leonid SHAMIS. « A SAT-based framework for efficient constrained clustering ». Dans *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 94–105. SIAM, 2010. 2.4.2, 2.4.2

[EB05] Niklas EÉN et Armin BIERE. « Effective preprocessing in SAT through variable and clause elimination ». Dans *International conference on theory and applications of satisfiability testing*, pages 61–75. Springer, 2005. 1.2.2, 11

[Een05] Niklas EEN. « MiniSat : A SAT solver with conflict-clause minimization ». Dans *Proc. SAT-05 : 8th Int. Conf. on Theory and Applications of Satisfiability Testing*, pages 502–518, 2005. 1, 2, 3

[EKSX96] Martin ESTER, Hans-Peter KRIEGEL, Jörg SANDER, et Xiaowei XU. « A density-based algorithm for discovering clusters in large spatial databases with noise. ». Dans *Kdd*, volume 96, pages 226–231, 1996. (document), 2, 2.3.3

[ES06] Niklas EÉN et Niklas SORENSSON. « Translating pseudo-boolean constraints into SAT ». *Journal on Satisfiability, Boolean Modeling and Computation*, 2 :1–26, 2006. 1.2.3, 1.2.3, 1.2.3, 1.2.3, 5.1

[FPRS07] Francois FOUSS, Alain PIROTTE, Jean-Michel RENDERS, et Marco SAERENS. « Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation ». *IEEE Transactions on knowledge and data engineering*, 19(3) :355–369, 2007. 2.1.5

[FVGG+14] Philippe FOURNIER-VIGER, Antonio GOMARIC, Ted GUENICHE, Azadeh SOLTANI, Cheng-Wei WU, et Vincent S TSENG. « SPMF : a java open-source pattern mining library ». *The Journal of Machine Learning Research*, 15(1) :3389–3393, 2014. 5.4.1

[GHSS07] Carla P GOMES, Joerg HOFFMANN, Ashish SABHARWAL, et Bart SELMAN. « From Sampling to Model Counting. ». Dans *IJCAI*, pages 2293–2299, 2007. 1.2.4, 4.2.1

[Gin93] Matthew L GINSBERG. « Dynamic backtracking ». *Journal of Artificial Intelligence Research*, 1 :25–46, 1993. 11

[GNDR11] Tias GUNS, Siegfried NIJSSEN, et Luc DE RAEDT. « Itemset mining : A constraint programming perspective ». *Artificial Intelligence*, 175(12-13) :1951–1983, 2011. 3.3.1

[GNDR13] Tias GUNS, Siegfried NIJSSEN, et Luc DE RAEDT. « k-Pattern set mining under constraints ». *IEEE Transactions on Knowledge and Data Engineering*, 25(2) :402–418, 2013. 3.3.2, 3.3.2, 3.3.2

[GOL79] Allen GOLDBERG. « On the Complexity of the Satisfiability Problem ». *Rapport n 16*, 1979. 11

[Gon85] Teofilo F GONZALEZ. « Clustering to minimize the maximum intercluster distance ». *Theoretical Computer Science*, 38 :293–306, 1985. 11

[GRS98] Sudipto GUHA, Rajeev RASTOGI, et Kyuseok SHIM. « CURE : an efficient clustering algorithm for large databases ». Dans *ACM Sigmod Record*, volume 27, pages 73–84. ACM, 1998. 6

[GSCK00] Carla P GOMES, Bart SELMAN, Nuno CRATO, et Henry KAUTZ. « Heavy-tailed phenomena in satisfiability and constraint satisfaction problems ». *Journal of automated reasoning*, 24(1-2) :67–100, 2000. 11

[GSK98] Carla P. GOMES, Bart SELMAN, et Henry KAUTZ. « Boosting Combinatorial Search Through Randomization ». Dans *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 431–437, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. 1

[GU89] Giorgio GALLO et Giampaolo URBANI. « Algorithms for testing the satisfiability of propositional formulae ». *The Journal of Logic Programming*, 7(1) :45 – 61, 1989. 1.2.1

[GZ03] Gösta GRAHNE et Jianfei ZHU. « Efficiently using prefix-trees in mining frequent itemsets. ». Dans *FIMI*, volume 90, 2003. 3.1.2

[Ham50] Richard W HAMMING. « Error detecting and error correcting codes ». *Bell Labs Technical Journal*, 29(2) :147–160, 1950. 2.1.2

[HFS+14] Lothar HOTZ, Alexander FELFERNIG, Markus STUMPTNER, Anna RYABOKON, Claire BAGLEY, et Katharina WOLTER. « *Configuration knowledge representation and reasoning* ». PhD thesis, Morgan Kaufmann, 2014. 4

[HHR10] Petr HÁJEK, Martin HOLENA, et Jan RAUCH. « The GUHA method and its meaning for data mining ». *Journal of Computer and System Sciences*, 76(1) :34 – 48, 2010. 5

[HJ97] Pierre HANSEN et Brigitte JAUMARD. « Cluster analysis and mathematical programming ». *Mathematical programming*, 79(1-3) :191–215, 1997. 2.2.1, 2.2.1

[HJS10] Youssef HAMADI, Saïd JABBOUR, et Lakhdar SAIS. « Learning for dynamic subsumption ». *International Journal on Artificial Intelligence Tools*, 19(04) :511–529, 2010. 1

[HK92] Lars HAGEN et Andrew B KAHNG. « New spectral methods for ratio cut partitioning and clustering ». *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9) :1074–1085, 1992. 2.2.2

[HLS02] Jin-Kao HAO, Frédéric LARDEUX, et Frédéric SAUBION. « A Hybrid Genetic Algorithm for the Satisfiability Problem ». Dans *1rst International Workshop on Heuristics*, 2002. 1.2.2

[HPY00] Jiawei HAN, Jian PEI, et Yiwen YIN. « Mining frequent patterns without candidate generation ». Dans *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000. 12

[HPYM04] Jiawei HAN, Jian PEI, Yiwen YIN, et Runying MAO. « Mining Frequent Patterns without Candidate Generation : A Frequent-Pattern Tree Approach ». *Data Mining and Knowledge Discovery*, 8(1) :53–87, 2004. 5

[Hua98] Zhexue HUANG. « Extensions to the k-means algorithm for clustering large data sets with categorical values ». *Data mining and knowledge discovery*, 2(3) :283–304, 1998. 11

[Hua07] Jinbo HUANG. « The Effect of Restarts on the Efficiency of Clause Learning. ». Dans *IJCAI*, volume 7, pages 2318–2323, 2007. 1

[HW79] John A HARTIGAN et Manchek A WONG. « Algorithm AS 136 : A k-means clustering algorithm ». *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1) :100–108, 1979. 8

[Jac12] Paul JACCARD. « The distribution of the flora in the alpine zone. ». *New phytologist*, 11(2) :37–50, 1912. 2.1.2, 4.2.1

[JLSS14] Said JABBOUR, Jerry LONLAC, Lakhdar SAIS, et Yakoub SALHI. « Extending modern SAT solvers for models enumeration ». Dans *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on*, pages 803–810. IEEE, 2014. 1.2.4, 5.4.1

[JMF99] Anil K JAIN, M Narasimha MURTY, et Patrick J FLYNN. « Data clustering : a review ». *ACM computing surveys (CSUR)*, 31(3) :264–323, 1999. 2.1.1

[JMRS17] Said JABBOUR, Nizar MHADHBI, Badran RADDAOUI, et Lakhdar SAIS. « A SAT-Based Framework for Overlapping Community Detection in Networks ». Dans *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part II*, pages 786–798, 2017. 3.3

[JRL08] Narendra JUSSIEN, Guillaume ROCHART, et Xavier LORCA. « Choco : an open source java constraint programming library ». Dans *CPAIOR'08 Workshop on Open-Source Software for Integer and Contraint Programming (OSSICP'08)*, pages 1–10, 2008. 1.3

[JSS13] Said JABBOUR, Lakhdar SAIS, et Yakoub SALHI. « The top-k frequent closed itemset mining using top-k sat problem ». Dans *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 403–418. Springer, 2013. 3.3.1

[JSS14] Said JABBOUR, Lakhdar SAIS, et Yakoub SALHI. « A Pigeon-Hole Based Encoding of Cardinality Constraints. ». Dans *ISAIM*, 2014. 1.2.3, 1.2.3, 5.1, 5.3.2

[JSS15a] Saïd JABBOUR, Lakhdar SAIS, et Yakoub SALHI. « Decomposition Based SAT Encodings for Itemset Mining Problems ». Dans *Proceedings of PAKDD'15*, pages 662–674, 2015. (document), 3.3.1, 3.3.1, 3.3.1, 4, 5.4.1

[JSS15b] Saïd JABBOUR, Lakhdar SAIS, et Yakoub SALHI. « On SAT Models Enumeration in Itemset Mining ». *CoRR*, abs/1506.02561, 2015.

[JW90] Robert G JEROSLOW et Jinchang WANG. « Solving propositional satisfiability problems ». *Annals of mathematics and Artificial Intelligence*, 1(1-4) :167–187, 1990. 11

[Kaz05] Przemysław KAZIENKO. « *Intelligent Information Processing and Web Mining : in Proceedings of IIPWM'05* », Chapitre IDARM — Mining of Indirect Association Rules, pages 77–86. Springer, 2005. 5.2.1

[Kaz09a] Przemysław KAZIENKO. « Mining indirect association rules for web recommendation ». *International Journal of Applied Mathematics and Computer Science*, 19(1) :165–186, 2009. (document), 3

[Kaz09b] Przemyslaw KAZIENKO. « Mining Indirect Association Rules for Web Recommendation ». *Applied Mathematics and Computer Science*, 19 :165–186, 2009. 5.2.1

[KBC11] Mehdi KHIARI, Patrice BOIZUMAULT, et Bruno CRÉMILLEUX. « A generic approach for modeling and mining n-ary patterns ». Dans *International Symposium on Methodologies for Intelligent Systems*, pages 300–305. Springer, 2011. 3.3.2, 3.3.2

[KHK99] George KARYPIS, Eui-Hong HAN, et Vipin KUMAR. « Chameleon : Hierarchical clustering using dynamic modeling ». *Computer*, 32(8) :68–75, 1999. 2, 6

[Kin67] Benjamin KING. « Step-wise clustering procedures ». *Journal of the American Statistical Association*, 62(317) :86–101, 1967. 7

[KR09] Leonard KAUFMAN et Peter J ROUSSEEUW. *Finding groups in data : an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009. 2.1.2

[KRD+17] Chia-Tung KUO, SS RAVI, Thi-Bich-Hanh DAO, Christel VRAIN, et Ian DAVIDSON. « A Framework for Minimal Clustering Modification via Constraint Programming. ». Dans *AAAI*, pages 1389–1395, 2017. 2.4.2

[Kry98a] Marzena KRYSZKIEWICZ. « Representative Association Rules ». Dans *Proceedings of PAKDD'98*, pages 198–209, 1998. 3.2.2, 5.2.4

[Kry98b] Marzena KRYSZKIEWICZ. « Representative Association Rules and Minimum Condition Maximum Consequence Association Rules ». Dans *Proceedings of PKDD '98*, pages 361–369, 1998. 5.2.3

[Kry02] Marzena KRYSZKIEWICZ. Concise representations of association rules. Dans *Pattern Detection and Discovery*, pages 92–109. Springer, 2002. 3.2.2, 3.2.2

[KV06] Ludmila I KUNCHEVA et Dmitry P VETROV. « Evaluation of stability of k-means cluster ensembles with respect to random initialization ». *IEEE transactions on pattern analysis and machine intelligence*, 28(11) :1798–1808, 2006. 4.2.1

[KW14] Stephen M KANG et Peter W WAGACHA. « Extracting diagnosis patterns in electronic medical records using association rule mining ». *International Journal of Computer Applications*, 108(15), 2014. (document), 3

[LA97] Chu Min LI et Anbulagan ANBULAGAN. « Heuristics based on unit propagation for satisfiability problems ». Dans *Proceedings of the 15th international joint conference on Artifical intelligence-Volume 1*, pages 366–371. Morgan Kaufmann Publishers Inc., 1997. 11

[LA99] Bjornar LARSEN et Chinatsu AONE. « Fast and effective text mining using linear-time document clustering ». Dans *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22. ACM, 1999. 2.1.4

[Lam91] Leslie LAMPORT. « The Temporal Logic of Actions ». Rapport technique 79, SRC, 1991.

[Lar92] Tracy LARRABEE. « Test pattern generation using Boolean satisfiability ». *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(1) :4–15, 1992. 11

[Lec13] Christophe LECOUTRE. *Constraint Networks : Targeting Simplicity for Techniques and Algorithms*. John Wiley & Sons, 2013. 1.3, 1.3

[LLL+16] Nadjib LAZAAR, Yahia LEBBAH, Samir LOUDNI, Mehdi MAAMAR, Valentin LEMIÈRE, Christian BESSIERE, et Patrice BOIZUMAULT. « A global constraint for closed frequent pattern mining ». Dans *International Conference on Principles and Practice of Constraint Programming*, pages 333–349. Springer, 2016. 3.3.1

[LS16] Xiaotong LIU et Han-Wei SHEN. « Association analysis for visual exploration of multivariate scientific data sets ». *IEEE transactions on visualization and computer graphics*, 22(1) :955–964, 2016. (document), 3

[LSZ93] Michael LUBY, Alistair SINCLAIR, et David ZUCKERMAN. « Optimal speedup of Las Vegas algorithms ». *Information Processing Letters*, 47(4) :173–180, 1993. 1

[Mac67] James MACQUEEN. « Some methods for classification and analysis of multivariate observations ». Dans *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. (document), 2, 2.1.1, 2.3.2

[Mar09] Frederic MARIS. « *Planification SAT et Planification Temporellement Expressive. Les Systemes TSP et TLP-GP.* ». PhD thesis, Université Paul Sabatier-Toulouse III, 2009. 1

[MBC09] Marcilio MENDONCA, Moises BRANCO, et Donald COWAN. « SPLOT : software product lines online tools ». Dans *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pages 761–762. ACM, 2009. 1

[MBC+12] Jean-Philippe MÉTIVIER, Patrice BOIZUMAULT, Bruno CRÉMILLEUX, Mehdi KHIARI, et Samir LOUDNI. « Constrained clustering using SAT ». Dans *International Symposium on Intelligent Data Analysis*, pages 207–218. Springer, 2012. 1

[McA80] David A MCALLESTER. « An Outlook on Truth Maintenance. ». Rapport technique, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1980. 11

[MD12] Vikas MARKAM et Lect Shirish Mohan DUBEY. « A general study of associations rule mining in intrusion detection system ». *International Journal of Emerging Technology and Advanced Engineering*, 2(1) :347–356, 2012. (document), 3

[MG04] Frank MITTLEBACH et Michel GOOSSENS. *The LaTeXCompanion (Second Edition)*. Addison Wesley Publishing Company, 2004.

[MK95] Jiri MATAS et Josef KITTLER. « Spatial and feature space clustering : Applications in image analysis ». Dans *International Conference on Computer Analysis of Images and Patterns*, pages 162–173. Springer, 1995. (document), 2

[MMZ+01] Matthew W MOSKEWICZ, Conor F MADIGAN, Ying ZHAO, Lintao ZHANG, et Sharad MALIK. « Chaff : Engineering an efficient SAT solver ». Dans *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001. 3

[Mor93] Paul MORRIS. « The Breakout Method for Escaping from Local Minima ». Dans *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI'93, pages 40–45. AAAI Press, 1993.

[MSL07] Joao MARQUES-SILVA et Inês LYNCE. « Towards robust CNF encodings of cardinality constraints ». Dans *International Conference on Principles and Practice of Constraint Programming*, pages 483–497. Springer, 2007. 1.2.3

[PB07] Dan PELLEG et Dorit BARAS. « K-means with large and noisy constraint sets ». Dans *European Conference on Machine Learning*, pages 674–682. Springer, 2007. 13

[PBTL99] Nicolas PASQUIER, Yves BASTIDE, Rafik TAOUIL, et Lotfi LAKHAL. « Discovering frequent closed itemsets for association rules ». Dans *International Conference on Database Theory*, pages 398–416. Springer, 1999. 3.1.2, 3.1.2, 3.1.2, 3.2.2

[PCY95] Jong Soo PARK, Ming-Syan CHEN, et Philip S YU. *An effective hash-based algorithm for mining association rules*, volume 24. ACM, 1995. 12

[PD09] Knot PIPATSRISAWAT et Adnan DARWICHE. « On the power of clause-learning SAT solvers with restarts ». Dans *International Conference on Principles and Practice of Constraint Programming*, pages 654–668. Springer, 2009. 1

[PG86] David A. PLAISTED et Steven GREENBAUM. « A Structure-preserving Clause Form Translation ». *J. Symb. Comput.*, 2(3) :293–304, septembre 1986. 1.1.3

[PHS08] Cédric PIETTE, Youssef HAMADI, et Lakhdar SAÏS. « Vivifying Propositional Clausal Formulae. ». Dans *ECAI*, volume 178, pages 525–529, 2008. 11

[PT98] Balaji PADMANABHAN et Alexander TUZHILIN. « A Belief-Driven Method for Discovering Unexpected Patterns ». pages 94–100. AAAI Press, 1998. 13, 13

[PT99] Balaji PADMANABHAN et Alexander TUZHILIN. « Unexpectedness as a measure of interestingness in knowledge discovery ». *Decision Support Systems*, 27(3) :303–318, 1999. (document), 3.2.2, 13

[RD00] Irina RISH et Rina DECHTER. « Resolution versus Search : Two Strategies for SAT ». *Journal of Automated Reasoning*, 24(1) :225–275, Feb 2000. 1.2.2

[RGN08] Luc De RAEDT, Tias GUNS, et Siegfried NIJSSEN. « Constraint Programming for Itemset Mining ». Dans *Proceedings of SIGKDD'08*, pages 204–212, 2008. (document), 3.3, 3.3.1, 3.3.1, 3.3.1, 5

[RNOH11] Luc De RAEDT, Siegfried NIJSSEN, Barry O'SULLIVAN, et Pascal Van HENTENRYCK. « Constraint Programming meets Machine Learning and Data Mining ». *Dagstuhl Reports*, 1(5) :61–83, 2011. (document), 5

[Rob65] John Alan ROBINSON. « A machine-oriented logic based on the resolution principle ». *Journal of the ACM (JACM)*, 12(1) :23–41, 1965. 1.2.2

[RS08] Vadim RYVCHIN et Ofer STRICHMAN. « Local restarts ». Dans *International Conference on Theory and Applications of Satisfiability Testing*, pages 271–276. Springer, 2008. 1

[RT60] David J ROGERS et Taffee T TANIMOTO. « A computer program for classifying plants ». *Science*, 132(3434) :1115–1118, 1960. 2.1.2

[SAG17] Pierre SCHAUS, John OR AOGA, et Tias GUNS. « CoverSize : a global constraint for frequency-based itemset mining ». Dans *International Conference on Principles and Practice of Constraint Programming*, pages 529–546. Springer, 2017. 3.3.1

[SB01] Sergio M SAVARESI et Daniel L BOLEY. « On the performance of bisecting K-means and PDDP ». Dans *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–14. SIAM, 2001. 6

[SF83] Alberto SANFELIU et King-Sun FU. « A distance measure between attributed relational graphs for pattern recognition ». *IEEE transactions on systems, man, and cybernetics*, (3) :353–362, 1983. 2.1.5

[Sin05] Carsten SINZ. « Towards an optimal CNF encoding of boolean cardinality constraints ». Dans *International conference on principles and practice of constraint programming*, pages 827–831. Springer, 2005. 1.2.3, 1.2.3, 1.2.3, 4.4, 5.2.3

[SKC93] Bart SELMAN, Henry A KAUTZ, et Bram COHEN. « Local search strategies for satisfiability testing. ». *Cliques, coloring, and satisfiability*, 26 :521–532, 1993. 1.2.2

[SLM92] Bart SELMAN, Hector J LEVESQUE, et David G MITCHELL. « A New Method for Solving Hard Satisfiability Problems. ». Dans *AAAI*, volume 92, pages 440–446, 1992. 1.2.2

[SNK06] Laszlo SZATHMARY, Amedeo NAPOLI, et Sergei O KUZNETSOV. « Zart : A multifunctional itemset mining algorithm ». 2006. 3.2.2, 5.4.1

[Sok63] Robert R SOKAL. « The principles and practice of numerical taxonomy ». *Taxon*, pages 190–199, 1963. 2.1.2

[SON95] Ashok SAVASERE, Edward Robert OMIECINSKI, et Shamkant B NAVATHE. « An efficient algorithm for mining association rules in large databases ». Rapport technique, Georgia Institute of Technology, 1995. 12

[SP05] Sathiamoorthy SUBBARAYAN et Dhiraj K. PRADHAN. « NiVER : Non-increasing Variable Elimination Resolution for Preprocessing SAT Instances ». Dans Holger H. HOOS et David G. MITCHELL, éditeurs, *Theory and Applications of Satisfiability Testing*, pages 276–291, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. 11

[SS62] Peter HA SNEATH et Robert R SOKAL. « Numerical taxonomy ». *Nature*, 193(4818) :855–860, 1962. 2, 7

[SS77] Richard M STALLMAN et Gerald J SUSSMAN. « Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis ». *Artificial intelligence*, 9(2) :135–196, 1977. 11

[SS96] JP Marques SILVA et Karem A SAKALLAH. « Conflict analysis in search algorithms for satisfiability ». Dans *Tools with Artificial Intelligence, 1996., Proceedings Eighth IEEE International Conference on*, pages 467–469. IEEE, 1996. 11

[SS97] Joao P Marques SILVA et Karem A SAKALLAH. « GRASP : a new search algorithm for satisfiability ». Dans *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, pages 220–227. IEEE Computer Society, 1997. 1.2.2

[STL10] Christian SCHULTE, Guido TACK, et Mikael Z LAGERKVIST. « Modeling and programming with gecode ». *Schulte, Christian and Tack, Guido and Lagerkvist, Mikael*, (2015), 2010. 1.3

[Suz02] Einoshin SUZUKI. « Undirected discovery of interesting exception rules ». *International Journal of Pattern Recognition and Artificial Intelligence*, 16(08) :1065–1086, 2002. (document), 3.2.2, 13

[SV94] Thomas SCHIEX et Gérard VERFAILLIE. « Nogood recording for static and dynamic constraint satisfaction problems ». *International Journal on Artificial Intelligence Tools*, 3(02) :187–207, 1994. 11

[Sza06] Laszlo SZATHMARY. « *Symbolic Data Mining Methods with the Coron Platform. (Méthodes symboliques de fouille de données avec la plate-forme Coron)* ». PhD thesis, Henri Poincaré University, Nancy, France, 2006. 3.2.2, 3.2.2, 3.2.2, 5.4.1

[TKS00] Pang-Ning TAN, Vipin KUMAR, et Jaideep SRIVASTAVA. « Indirect Association : Mining Higher Order Dependencies in Data ». Dans *Proceedings of PKDD'00*, pages 632–637, 2000. 5, 5.2

[TKS02] Pang-Ning TAN, Vipin KUMAR, et Jaideep SRIVASTAVA. « Selecting the Right Interestingness Measure for Association Patterns ». Dans *Proceedings of SIGKDD'02*, pages 32–41, 2002. 5.2.1

[TPBL00] Rafik TAOUIL, Nicolas PASQUIER, Yves BASTIDE, et Lotfi LAKHAL. « Mining Bases for Association Rules Using Closed Sets ». Dans *Proceedings of ICDE'00*, page 307, 2000. 5, 5.2

[Tse68] G.S. TSEITIN. « On the complexity of derivations in the propositional calculus ». Dans H.A.O. SLESENKO, éditeur, *Structures in Constructives Mathematics and Mathematical Logic, Part II*, pages 115–125, 1968. 1.1.3, 1.2.3

[TTBHCKC16] Guns TIAS, Dao THI-BICH-HANH, l Vrain CHRISTE, et Duong KHANH-CHUONG. « Repetitive branch-and-bound using constraint programming for constrained minimum sum-of-squares clustering ». Dans *ECAI 2016 : 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands-Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285, page 462. IOS Press, 2016. 2.4.2

[Tve77] Amos TVERSKY. « Features of similarity. ». *Psychological review*, 84(4) :327, 1977. 4.2.1

[Tve04] Amos TVERSKY. *Preference, belief, and similarity : selected writings*. MIT Press, 2004. 4.2.1

[UKA04] Takeaki UNO, Masashi KIYOMI, et Hiroki ARIMURA. « LCM ver. 2 : Efficient mining algorithms for frequent/closed/maximal itemsets ». Dans *Fimi*, volume 126, 2004. 3.1.2

[VB03] Miroslav N VELEV et Randal E BRYANT. « Effective use of boolean satisfiability procedures in the formal verification of superscalar and vliw microprocessors ». *Journal of Symbolic Computation*, 35(2) :73–106, 2003. 1

[War98] Joost P WARNERS. « A linear-time transformation of linear inequalities into conjunctive normal form ». *Information Processing Letters*, 68(2) :63–69, 1998. 1.2.3, 5.1

[WC00] Kiri WAGSTAFF et Claire CARDIE. « Clustering with Instance-level Constraints ». Dans *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 1103–1110, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. 2.4

[WCRS01] Kiri WAGSTAFF, Claire CARDIE, Seth ROGERS, et Stefan SCHRÖDL. « Constrained k-means clustering with background knowledge ». Dans *ICML*, volume 1, pages 577–584, 2001. 2.4.1

[WEL+10] Tobias WITTKOP, Dorothea EMIG, Sita LANGE, Sven RAHMANN, Mario ALBRECHT, John H MORRIS, Sebastian BÖCKER, Jens STOYE, et Jan BAUMBACH. « Partitioning biological data with transitivity clustering ». *Nature methods*, 7(6) :419, 2010. (document), 2

[WJ63] Joe H WARD JR. « Hierarchical grouping to optimize an objective function ». *Journal of the American statistical association*, 58(301) :236–244, 1963. (document)

[WXL99] Ke WANG, Chu XU, et Bing LIU. « Clustering transactions using large items ». Dans *Proceedings of the eighth international conference on Information and knowledge management*, pages 483–490. ACM, 1999. (document), 2

[XJRN03] Eric P XING, Michael I JORDAN, Stuart J RUSSELL, et Andrew Y NG. « Distance metric learning with application to clustering with side-information ». Dans *Advances in neural information processing systems*, pages 521–528, 2003. 13

[XW08] Rui XU et Don WUNSCH. *Clustering*, volume 10. John Wiley & Sons, 2008. 2.1, 2.1, 2.1.1, 2.1.2

[XYFS07] Xiaowei XU, Nurcan YURUK, Zhidan FENG, et Thomas AJ SCHWEIGER. « Scan : a structural clustering algorithm for networks ». Dans *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833. ACM, 2007. 2.3.4

[YFM+01] Ka Yee YEUNG, Chris FRALEY, Alejandro MURUA, Adrian E. RAFTERY, et Walter L. RUZZO. « Model-based clustering and data transformations for gene expression data ». *Bioinformatics*, 17(10) :977–987, 2001. 8

[Zak00] Mohammed Javeed ZAKI. « Scalable algorithms for association mining ». *IEEE transactions on knowledge and data engineering*, 12(3) :372–390, 2000. 12

[Zak04] Mohammed Javeed ZAKI. « Mining Non-Redundant Association Rules ». *Data Mining Knowledge Discovery*, 9 :223–248, 2004. (document), 3.1.2, 3.2.2, 3.2.2, 3.2.2, 5, 5.2, 5.2.4, 5.2

[ZH02] Mohammed J ZAKI et Ching-Jui HSIAO. « CHARM : An efficient algorithm for closed itemset mining ». Dans *Proceedings of the 2002 SIAM international conference on data mining*, pages 457–473. SIAM, 2002. 3.1.2

[ZMMM01] Lintao ZHANG, Conor F MADIGAN, Matthew H MOSKEWICZ, et Sharad MALIK. « Efficient conflict driven learning in a boolean satisfiability solver ». Dans *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 279–285. IEEE Press, 2001. 11, 11, 2, 3

[Zub38] Joseph ZUBIN. « A technique for measuring like-mindedness. ». *The Journal of Abnormal and Social Psychology*, 33(4) :508, 1938. 2.1.2

Résumé

Dans cette thèse, nous abordons les problèmes bien connus de clustering et de fouille de règles d'association. Notre première contribution introduit un nouveau cadre de clustering, où les objets complexes sont décrits par des formules propositionnelles. Premièrement, nous adaptons les deux fameux algorithmes de clustering, à savoir, le k-means et l'algorithme hiérarchique ascendant, pour traiter ce type d'objets complexes. Deuxièmement, nous introduisons un nouvel algorithme hiérarchique descendant pour le clustering des objets représentés explicitement par des ensembles de modèles. Enfin, nous proposons un encodage basé sur la satisfiabilité propositionnelle du problème de clustering des formules propositionnelles sans avoir besoin d'une représentation explicite de leurs modèles. Dans une seconde contribution, nous proposons une nouvelle approche basée sur la satisfiabilité pour extraire les règles d'association en une seule étape. La tâche est modélisée comme une formule propositionnelle dont les modèles correspondent aux règles à extraire. Pour montrer la flexibilité de notre cadre, nous abordons également d'autres variantes, à savoir, l'extraction des règles d'association fermées, minimales non redondantes, les plus générales et les indirectes. Les expérimentations sur de nombreux jeux de données montrent que sur la majorité des tâches de fouille de règles d'association considérées, notre approche déclarative réalise de meilleures performances que les méthodes spécialisées.

Mots-clés: fouille de données, clustering, règles d'association, satisfiabilité propositionnelle.

Abstract

In this thesis, We adress the well-known clustering and association rules mining problems. Our first contribution introduces a new clustering framework, where complex objects are described by propositional formulas. First, we extend the two well-known k-means and hierarchical agglomerative clustering techniques to deal with these complex objects. Second, we introduce a new divisive algorithm for clustering objects represented explicitly by sets of models. Finally, we propose a propositional satisfiability based encoding of the problem of clustering propositional formulas without the need for an explicit representation of their models. In a second contribution, we propose a new propositional satisfiability based approach to mine association rules in a single step. The task is modeled as a propositional formula whose models correspond to the rules to be mined. To highlight the flexibility of our proposed framework, we also address other variants, namely the closed, minimal non-redundant, most general and indirect association rules mining tasks. Experiments on many datasets show that on the majority of the considered association rules mining tasks, our declarative approach achieves better performance than the state-of-the-art specialized techniques.

Keywords: Data mining, Clustering, Association rules, Propositional satisfiability.