Clustering conceptuel et relationnel en programmation par contraintes

Thi-Bich-Hanh Dao, Willy Lesaint, Christel Vrain

Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, F-45067, Orléans, France
{thi-bich-hanh.dao, willy.lesaint, christel.vrain}@univ-orleans.fr

Résumé

Différentes tâches de clustering existent dans le domaine de la fouille de données. Le clustering conceptuel pour les données qualitatives vise à identifier des concepts, les clusters sont alors définis par ces concepts. Le clustering relationnel cherche à partitionner les données dans des clusters homogènes et/ou bien séparés à partir d'une mesure de dissimilarité entre les objets. Des travaux récents ont proposé des cadres généraux et déclaratifs fondés sur la programmation par contraintes pour le clustering conceptuel ou pour le clustering relationnel. Dans ce papier, nous proposons un cadre plus général qui unifie les clusterings conceptuel et relationnel en programmation par contraintes. Ce cadre permet de tirer avantage des deux approches, en optimisant un critère relationnel ou conceptuel et en intégrant des contraintes qu'elles soient de structure, qu'elles portent sur les distances entre les objets ou sur leurs propriétés communes. Nous proposons de plus un modèle amélioré pour traiter les contraintes conceptuelles reposant sur des contraintes ensemblistes.

1 Introduction

Depuis une dizaine d'années, il a été montré que des formalismes déclaratifs, comme la Programmation par Contraintes, permettaient de modéliser facilement diverses tâches de fouille de données, offrant ainsi une grande flexibilité pour modéliser le problème et intégrer des connaissances utilisateur. En particulier, plusieurs approches se sont intéressées à la classification non supervisée, appelée par la suite clustering, dont le but est de regrouper des objets en classes (clusters) de façon à ce que les objets d'un même cluster soient les plus similaires possibles. On peut distinguer deux approches : le clustering relationnel [2, 3, 4] et le clustering conceptuel [11, 15]. La première, certainement la plus connue, repose sur une mesure de dissimilarité entre les objets, en général la distance euclidienne lorsque les données sont décrites par des attributs numériques. Le regroupement souhaité est souvent modélisé par un critère d'optimisation, comme par exemple minimiser la dissimilarité maximale des objets d'un même cluster, ou maximiser la marge minimale entre les clusters. Cependant, l'interprétation des clusters est difficile, puisque l'importance des différents attributs est agglomérée dans la mesure de dissimilarité.

Le clustering conceptuel s'applique sur des objets décrits par des attributs qualitatifs, souvent booléens. Il vise à regrouper les objets en clusters, de telle manière que chaque cluster puisse être identifié par un ensemble d'attributs communs à tous les objets du cluster. Dans ce cadre, les attributs quantitatifs (numériques) doivent être, dans une étape préalable au clustering, discrétisés en attributs qualitatifs. L'un des intérêts de cette approche est de trouver des clusters qui sont alors des couples formés d'un ensemble d'objets et d'un ensemble d'attributs décrivant ces objets. La discrétisation influence fortement le résultat.

Les méthodes conceptuelles semblent donc plus adaptées aux données symboliques, même s'il est possible de discrétiser des attributs numériques, mais elles ont l'avantage de donner une interprétation des clusters en terme des propriétés que leurs objets vérifient. Quant aux approches relationnelles, elles nécessitent de définir une mesure de dissimilarité entre les objets, ce qui est moins naturel lorsque les données sont hétérogènes, composées de descripteurs quantitatifs et qualitatifs. De plus, elles ne permettent pas d'intégrer directement des contraintes sur les propriétés vérifiées par les objets d'un même cluster.

Nous présentons dans ce papier un cadre général et déclaratif, fondé sur la Programmation par Contraintes, permettant d'exploiter les avantages des deux approches : les données peuvent être décrites

par des attributs quantitatifs ou qualitatifs ; à ces données, sont associés une mesure de dissimilarité calculée sur tout ou partie des attributs et un sous-ensemble des attributs qualitatifs. Il est alors possible d'intégrer des contraintes relationnelles, fondées sur les distances entre les objets et/ou des contraintes conceptuelles fondées sur les propriétés des objets des clusters. Le critère d'optimisation peut lui aussi être relationnel ou conceptuel. Notons que notre approche, lorsqu'appliquée sans contrainte conceptuelle, modélise le clustering relationnel et vice versa. Ce cadre étend donc le clustering relationnel en PPC, introduit par [2, 3, 4] et le clustering conceptuel, présenté dans [11, 15].

De plus, nous proposons dans ce papier deux réalisations des contraintes conceptuelles : la première est l'implantation directe du modèle proposé dans [11], la seconde repose sur des contraintes ensemblistes et nous montrons dans nos expérimentations que cette dernière est plus efficace en termes de propagation que la première. Enfin, nous montrons sur un jeu de données l'intérêt de coupler les approches relationnelle et conceptuelle.

Le papier est organisé comme suit. La section 2 présente le clustering conceptuel, le clustering relationnel et les travaux modélisant ces deux types de clustering dans des approches déclaratives. La section 3 présente notre cadre, intégrant clustering relationnel et conceptuel ; deux réalisations de la vue conceptuelle sont présentées. Enfin, la section 4 est dédiée aux expérimentations.

2 Clustering conceptuel et relationnel sous contraintes utilisateur

Soit $\mathcal{O} = \{o_1, ..., o_n\}$ un ensemble de n objets. Le clustering vise à regrouper les données en un ensemble de clusters homogènes. Le problème est souvent formulé comme la recherche d'une partition en k clusters $\mathcal{C} = \{C_1, .., C_k\}$ satisfaisant les conditions fixées par l'utilisateur et éventuellement, optimisant un critère donné. Pour chaque cluster $c \in \mathcal{C}$, on note O_c l'ensemble des objets du cluster. Les clusters forment une partition, i.e. (1) pour tout $c \in \mathcal{C}$, $O_c \neq \emptyset$, (2) $\cup_c O_c = \mathcal{O}$ et (3) pour tout $c \neq c'$, $O_c \cap O_{c'} = \emptyset$. On distingue en général le clustering conceptuel et le clustering relationnel.

2.1 Clustering conceptuel sous contraintes utilisateur

Dans le clustering conceptuel, les objets sont décrits par un ensemble d'attributs booléens, aussi appelés propriétés, $\mathcal{A} = \{a_1, ..., a_m\}$. Les données sont représentées par une matrice binaire t de taille $n \times m$ telle que $\forall o \in \mathcal{O}, \forall a \in \mathcal{A}$, $t_{oa} = 1$ si et seulement si l'objet o vérifie l'attribut a. Un cluster est alors vu comme un couple formé d'un ensemble d'objets constituant le cluster et d'un ensemble d'attributs caractérisant les objets de ce cluster. Un cluster doit en général vérifier une propriété de fermeture, à savoir les objets du cluster satisfont tous les attributs caractérisant ce cluster (propriété d'intention) et seuls ces objets les vérifient (propriété d'extension). Un cluster satisfaisant cette propriété de fermeture est alos appelé un concept. Pour chaque cluster $c \in \mathcal{C}$, on note A_c l'ensemble des attributs caractérisant le cluster.

Considérons par exemple les données suivantes :

	p_1	p_2	p_3	p_4
o_1	1	1	1	0
o_2	1	1	0	0
o_3	1	1	0	1
o_4	1	0	1	1
o_5	0	1	1	1

Le cluster composé des objets $\{o_1, o_2\}$ satisfait les propriétés p_1 et p_2. Mais il n'est pas fermé, puisque o_3 satisfait aussi les propriétés p_1 et p_2. En revanche les couples $(\{o_1, o_2, o_3\}, \{p_1, p_2\})$ et $(\{o_4, o_5\}, \{p_3, p_4\})$ vérifient la propriété de fermeture et sont des concepts.

Des contraintes et des critères d'optimisation modélisent les conditions qui doivent être satisfaites par les clusters [11, 10].

Contrainte de taille des clusters, aussi appelée contrainte de fréquence. Elle permet de contraindre le nombre d'objets contenus dans chaque cluster par une borne minimale ou maximale, notée b : $\forall c \in \mathcal{C}, |O_c| \lesseqgtr b$.

Contrainte de taille des concepts. La taille d'un cluster, vu comme un concept, est le nombre d'attributs caractérisant ce cluster. On peut de la même manière contraindre ce nombre par une borne b : $\forall c \in \mathcal{C}, |A_c| \lesseqgtr b$.

Contrainte d'extension. L'extension d'un ensemble d'attributs A, $A \subseteq \mathcal{A}$ est l'ensemble des objets vérifiant ces attributs :
$\text{ext}(A) = \{o \in \mathcal{O} \mid \forall a \in A, t_{oa} = 1\}$.

La contrainte d'extension est appliquée à chaque cluster ; elle assure que l'extension des attributs de ce cluster est exactement l'ensemble des objets de ce cluster : $\forall c \in \mathcal{C},\ O_c = \text{ext}(A_c)$.

Notons que $O_c \subseteq \text{ext}(A_c)$ assure que chaque objet du cluster c vérifie tous les attributs du cluster et que $\text{ext}(A_c) \subseteq O_c$ assure qu'aucun autre objet de $\mathcal{O}$ ne possède tous ces attributs.

Etant donné que l'on se place dans le cadre d'une partition des objets, $\text{ext}(A_c) \subseteq O_c$ assure aussi qu'aucun autre cluster ne contient dans son intention cet ensemble d'attributs (sinon ce cluster serait inclus dans

ext(A_c) et donc dans O_c). Avec cette contrainte, dans ce cadre du clustering conceptuel par partionnement, il est alors impossible d'avoir deux clusters dont l'ensemble des attributs de l'un est inclus dans l'ensemble des attributs de l'autre.

Contrainte d'intention. L'intention d'un ensemble d'objets O, $O \subseteq \mathcal{O}$, est l'ensemble des attributs communs à ces objets :
$\text{int}(O) = \{a \in \mathcal{A} \mid \forall o \in O,\ t_{oa} = 1\}$.

La contrainte d'intention est appliquée à tous les clusters, elle assure pour chaque cluster $c \in \mathcal{C}$ que l'intention des objets de c est exactement l'ensemble des attributs de c : $\forall c \in \mathcal{C}, A_c = \text{int}(O_c)$.

De même que précédemment, notons que $A_c \subseteq \text{int}(O_c)$ assure que chaque objet de c possède bien les attributs A_c du cluster et $\text{int}(O_c) \subseteq A_c$ assure que toutes les propriétés communes à O_c appartiennent bien aux attributs du cluster.

Contrainte de fermeture. La contrainte de fermeture est la conjonction des contraintes d'intention et d'extension : $\forall c \in \mathcal{C},\ O_c = \text{ext}(A_c)\ \wedge\ A_c = \text{int}(O_c)$.

Elle assure que (O_c, A_c) est un concept au sens de l'Analyse Formelle de Concepts.

Maximiser le nombre d'objets par cluster.

Afin d'équilibrer la taille des clusters, on peut choisir d'avoir le plus d'objets possibles dans le cluster de plus petite taille : maximiser($\min\{|O_c| \mid c \in \mathcal{C}\}$).

Maximiser le nombre d'attributs caractérisant les clusters.

De la même façon, on peut choisir d'optimiser la taille du concept minimal des clusters : maximiser($\min\{|A_c| \mid c \in \mathcal{C}\}$).

2.2 Clustering relationnel sous contraintes utilisateur

On suppose qu'il existe une mesure de dissimilarité entre les objets et on note $d(o, o')$ la dissimilarité entre deux objets $o, o' \in \mathcal{O}$. Cette dissimilarité est calculée à partir des attributs en utilisant, par exemple, la distance euclidienne. Le clustering relationnel regroupe les objets en optimisant un critère fondé sur la dissimilarité entre les objets. Le critère d'optimisation peut être, par exemple, minimiser le diamètre maximal des clusters ou maximiser la marge minimale entre clusters.

Minimiser le diamètre. Minimiser le diamètre consiste à minimiser le diamètre du cluster le plus grand, c'est-à-dire à minimiser la plus grande distance entre deux objets d'un même cluster :

minimiser($\max\{d(o, o') \mid c \in \mathcal{C} \wedge o \in O_c \wedge o' \in O_c\}$).

Notons que minimiser le diamètre a pour effet de séparer dans des clusters différents les objets éloignés.

Maximiser la marge. Maximiser la marge propose une vision duale de l'optimisation précédente. En effet, on souhaite ici différencier au maximum les clusters, en maximisant la distance entre des points situés dans des clusters différents.

maximiser($min\{d(o, o') \mid c \neq c' \in \mathcal{C}, o \in O_c, o' \in O_{c'}\}$).

Des contraintes utilisateur sont également intégrées dans les tâches de clustering relationnel. Elles peuvent être des contraintes sur des objets (must-link, cannot-link) [17, 18] ou des contraintes sur des clusters [5, 6].

Contraintes must-link et cannot-link. Une contrainte must-link sur deux objets o, o' indique qu'ils doivent être dans le même cluster : $\forall c \in \mathcal{C},\ o \in O_c \Leftrightarrow o' \in O_c$. A l'inverse, une contrainte cannot-link indique que deux objets ne doivent pas être dans le même cluster : $\forall c \in \mathcal{C},\ \neg(o \in O_c \wedge o' \in O_c)$.

Contrainte de taille des clusters. Cette contrainte est identique à la contrainte du même nom (ou contrainte de fréquence) dans le clustering conceptuel. Elle permet de fixer une valeur maximale ou minimale sur le nombre d'objets dans chaque cluster.

Contrainte de diamètre La contrainte de diamètre permet de fixer une borne supérieure b au diamètre des clusters, i.e., à la distance maximale entre les objets du cluster : $\forall c \in \mathcal{C}, \forall o, o' \in O_c,\ d(o, o') \leq b$.

Contrainte de marge. On peut également contraindre la distance entre les clusters en imposant que cette distance soit supérieure à un seuil b : $\forall c \neq c' \in \mathcal{C}, \forall o \in O_c, \forall o' \in O_{c'},\ d(o, o') \geq b$.

2.3 Travaux connexes

Depuis quelques années, différents travaux investissent les cadres génériques et déclaratifs pour différentes tâches de clustering sous contraintes utilisateur. Ces cadres sont développés à base de la programmation par contraintes, de la programmation linéaire sur des entiers [16, 9, 1] ou de la programmation quadratique [19]. L'objectif général est de proposer un cadre où l'utilisateur peut intégrer différents types de contraintes.

Concernant le clustering conceptuel, les premiers travaux modélisent la recherche d'itemsets (ensembles d'attributs) dans le cadre de la programmation par contraintes [8, 10], de SAT [15, 12] ou d'ASP [13]. La recherche d'itemsets est généralisée en recherche d'ensembles d'itemsets, formalisant ainsi le clustering conceptuel. Dans [11], les auteurs reformulent les problèmes de recherche d'ensembles d'items en terme de programmation par contraintes. Ils formalisent ainsi la plupart des contraintes classiques de fouille de données. Le cadre déclaratif et très général de la programmation par contrainte permet alors de s'attaquer à la

plupart des problèmes de recherche d'itemsets. Le clustering conceptuel est formulé par la recherche d'un ensemble d'itemsets satisfaisant des contraintes spécifiques [11]. Dans cette lignée, un langage basé sur des contraintes a été proposé [15]. Des tâches de clustering conceptuel peuvent être formulées par des requêtes dans ce langage, qui sont ensuite traduites par des clauses SAT puis résolues par un solveur SAT.

Quant au clustering relationnel à base de dissimilarités, plusieurs cadres génériques sont développés permettant à l'utilisateur d'intégrer différents types de contraintes. Nous pouvons citer par exemple l'approche à base de programmation linéaire sur des entiers pour le critère de la somme des carrés intra-cluster [1] ou l'approche à base de SAT pour le cas de partition en 2 clusters ($k = 2$) [7]. Basée sur la programmation par contraintes, un cadre général et déclaratif a été proposé [2, 3, 4]. Ce cadre offre à l'utilisateur le choix de différents critères d'optimisation et des combinaisons de différents types de contraintes utlisateur.

3 Modélisation par la programmation par contraintes

Nous présentons un cadre en programmation par contraintes, qui unifie le clustering conceptuel et le clustering relationnel sous contraintes utilisateur. Ce cadre permet de traiter des données hétérogènes, qui peuvent être caractérisées par des attributs numériques (quantitatifs) et des attributs symboliques (qualitatifs). L'utilisateur peut choisir parmi les attributs ceux qui seront étudiés comme propriétés des objets et ceux utilisés pour calculer une dissimilarité entre paires d'objets. Il peut ensuite poser des contraintes conceptuelles sur des propriétés ainsi que des contraintes relationnelles fondées sur la dissimilarité. Il peut choisir de trouver toutes les solutions ou de trouver une solution optimale, étant donné un critère d'optimisation, pouvant relever du clustering conceptuel ou du clustering relationnel. Le cadre complet permet donc plusieurs possibilités :

- il peut se décliner en des tâches du clustering purement conceptuel ou purement relationnel,
- il permet de réaliser des tâches du clustering conceptuel (ou relationnel) en intégrant des contraintes relationnelles (ou conceptuelles, respectivement).

Nous présentons ce cadre en quatre parties : contraintes liées à la structure de l'ensemble de clusters, contraintes relationnelles, contraintes conceptuelles et critères d'optimisation. Le modèle est basé sur le modèle du clustering relationnel sous contraintes utilisateur [3, 4]. Nous présentons deux réalisations pour les contraintes conceptuelles : une réalisation binaire qui est une extension directe des travaux de [8, 10] et une réalisation utilisant des contraintes ensemblistes.

3.1 Contraintes sur la structure

Dans ce qui suit, on considère un ensemble $\mathcal{O}$ de n objets décrits sur un ensemble d'attributs booléens $\mathcal{A}$ de m attributs. Sans perdre de généralité, nous supposons que les objets et les attributs sont indexés et identifiés par leur indice, qui est $o \in [1, n]$ [1] pour les objets et $a \in [1, m]$ pour les attributs. Ceci implique aussi que $\mathcal{O}$ (ou $\mathcal{A}$) peut être utilisé pour l'ensemble $[1, n]$ (ou $[1, m]$, resp.) ou vice versa. La description des objets est représentée par une matrice binaire t de taille $n \times m$ telle que $\forall o \in \mathcal{O}, \forall a \in \mathcal{A}$, $t_{oa} = 1$ si et seulement si l'objet o vérifie l'attribut a. De plus, une mesure de dissimilarité d permet de calculer une dissimilarité entre toute paire d'objets.

Chaque cluster est identifié par son numéro, qui varie de 1 à k. Soit $\mathcal{C}$ l'ensemble des clusters $[1, k]$. Pour représenter l'appartenance de chaque objet $o \in [1, n]$ aux clusters, nous utilisons, comme dans [3], des variables $G : \mathcal{O} \to \mathcal{C}$, telles que $G[o] = c$ signifie que l'objet o appartient au cluster c. Le domaine des variables $G[o]$ est l'ensemble des entiers $[1, k]$.

Contraintes de partition Le choix des variables G représente naturellement une partition. Cependant il peut engendrer des solutions symétriques. Par exemple, deux solutions dans lesquelles deux clusters sont intervertis sont symétriques. Les objets composant les clusters sont les mêmes mais les numéros des clusters diffèrent. Supprimer ces symétries se fait par l'ajout de contraintes. Une stratégie consiste à numéroter les clusters dans l'ordre croissant. Plus précisément, le premier objet est placé dans le cluster numéro 1 puis les suivants sont placés peu à peu soit dans un cluster déjà créé, soit dans un nouveau cluster. Le numéro donné au nouveau cluster doit être égal au plus grand numéro déjà utilisé plus 1. Si l'on considère le tableau G, on doit avoir $G[1] = 1$ et pour toute valeur $c \in [2, k]$, s'il existe un indice i tel que $G[i] = c$, alors il doit exister un indice $j < i$ tel que $G[j] = c - 1$. Ces conditions sont exprimées par la contrainte $\mathit{precede}(G, [1, \ldots, k])$ [14].

Pour assurer d'obtenir à la fin k clusters non vides, la valeur k doit apparaître au moins une fois dans le tableau G. Ceci est assuré par la contrainte : $\mathit{atleast}(1, G, k)$.

1. Nous utilisons la notation $[1, n]$ pour désigner l'ensemble des entiers $\{1, .., n\}$.

Contraintes must-link et cannot-link Une contrainte must-link sur deux objets o, o' se traduit facilement par : $G[o] = G[o']$. De la même façon, une contrainte cannot-link se traduit par : $G[o] \neq G[o']$.

Contrainte de taille de clusters Le fait que chaque cluster c doit avoir au moins (ou au plus) b objets revient à ce que la valeur c apparaisse au moins (resp. au plus) b fois dans le tableau G. Ceci est représenté par la contrainte $atleast(b, G, c)$ (resp. $atmost(b, G, c)$), pour toute valeur $c \in \mathcal{C}$.

3.2 Contraintes relationnelles

Le diamètre maximal des clusters et la marge minimale entre clusters sont représentés respectivement par des variables D et S. Leur domaine est l'intervalle formé par la distance minimale et la distance maximale entre deux objets.

Contrainte de diamètre Une contrainte de diamètre fixe une borne supérieure $d_{\max}$ au diamètre des clusters. Tous les objets $o, o' \in \mathcal{O}$ tels que $d(o, o') > d_{\max}$ doivent être placés dans des clusters différents. Donc pour tout $o, o' \in \mathcal{O}$, si $d(o, o') > d_{\max}$, on pose une contrainte $G[o] \neq G[o']$.

Contrainte de marge Une contrainte de marge fixe une borne inférieure $d_{\min}$ à la marge entre clusters. Tous les objets $o, o' \in \mathcal{O}$ tels que $d(o, o') < d_{\min}$ doivent être placés dans le même cluster. Donc pour tout $o, o' \in \mathcal{O}$, si $d(o, o') < d_{\min}$, on pose une contrainte $G[o] = G[o']$.

3.3 Contraintes conceptuelles

Chaque cluster, composé d'un ensemble d'objets, est aussi caractérisé par un ensemble d'attributs. Pour représenter les attributs associés aux clusters et modéliser les contraintes conceptuelles permettant d'exprimer des conditions sur ces propriétés, nous présentons deux modèles différents. Le premier modèle utilise celui de [10, 11], à base de variables binaires. Le deuxième modèle repose sur des variables ensemblistes.

3.3.1 Modèle à base de variables binaires

Pour représenter les attributs associés aux clusters, de même que dans le modèle de [11], nous utilisons des variables binaires $A : \mathcal{C} \times \mathcal{A} \to \{0, 1\}$, où $A[c, a] = 1$ signifie que l'attribut a participe à A_c, l'ensemble qui caractérise le cluster c. Cela nécessite $k \times m$ variables binaires.

Contrainte de taille de concepts Elle peut se représenter par :

$$\forall c \in \mathcal{C}, \sum_{a \in \mathcal{A}} A[c, a] \lesseqgtr b \tag{1}$$

où b est la borne de taille. Cette contrainte est réalisée par k contraintes de somme linéaire sur des variables de A.

Contrainte d'extension Pour tout $o \in \mathcal{O}$, pour tout $c \in \mathcal{C}$:

$$G[o] = c \Leftrightarrow \sum_{1 \leq a \leq m} A[c, a](1 - t_{oa}) = 0 \tag{2}$$

Cette contrainte est réalisée par $n \times k$ contraintes réifiées, chacune est liée à une contrainte de somme linéaire sur des variables de A.

Contrainte d'intention Pour tout $c \in \mathcal{C}$, pour tout $a \in \mathcal{A}$:

$$A[c, a] \Leftrightarrow \sum_{1 \leq o \leq n} (G[o] = c)(1 - t_{oa}) = 0 \tag{3}$$

Cette contrainte est réalisée par $m \times k$ contraintes réifiées, chacune est liée à une contrainte de somme linéaire sur des variables booléennes représentant $G[o] = c$.

Contrainte de fermeture Par sa définition, la contrainte de fermeture sur $\mathcal{C}$ sera réalisée par les contraintes (2) et (3), qui représentent respectivement l'extension et l'intention des clusters de $\mathcal{C}$.

3.3.2 Modèle utilisant des variables ensemblistes

Nous proposons d'utiliser des variables ensemblistes $E : \mathcal{C} \to \mathcal{P}(\mathcal{A})$ pour représenter les ensembles d'attributs associés aux clusters. Cela nécessite donc k variables ensemblistes E_c, où chaque variable représente un ensemble $E_c \subseteq \mathcal{A}$.

De la même manière, la représentation des données sur les attributs qualitatifs est modifiée : chaque objet $o \in \mathcal{O}$ est caractérisé par l'ensemble t_o des attributs qu'il vérifie.

Avec cette modélisation, la réalisation des contraintes suivantes est modifiée.

La contrainte de taille de concept (1) devient :

$$\forall c \in \mathcal{C},\ |E[c]| \lesseqgtr b$$

qui est représentée par une contrainte *cardinality* pour chaque $c \in \mathcal{C}$.

La contrainte extension (2) devient

$$\forall c \in \mathcal{C}, \forall o \in \mathcal{O},\ G[o] = c \Leftrightarrow E_c \subseteq t_o$$

Cette contrainte est réalisée par $n \times k$ contraintes réifiées, chacune est liée à une contrainte d'inclusion.

La contrainte intention (3) devient

$$\forall c \in \mathcal{C},\ E_c = \cap_{G[o]=c} t_o$$

Elle est réalisée par k contraintes $E_c = \cap_{G[o]=c} t_o$, avec $c \in \mathcal{C}$. Chacune de ces contraintes nécessite n contraintes de domaine *dom* réifiées pour construire l'ensemble $I_c = \{o \in \mathcal{O} \mid G[o] = c\}$ et une contrainte *element* ensembliste effectuant $E_c = \cap(\langle t_1, ..., t_n \rangle [I_c])$.

3.4 Critères d'optimisation

Minimiser le diamètre Pour représenter le diamètre maximal des clusters nous utilisons une variable D du domaine des nombres flottants. Par conséquent, si deux objets o, o' ont une dissimilarité plus grande que D, ils doivent être dans des clusters différents. Cela est représenté par des contraintes réifiées. Nous posons donc pour chaque paire $o < o' \in \mathcal{O}$ la contrainte réifiée :

$$d(o, o') > D \Rightarrow G[o] \neq G[o'].$$

Lorsque l'utilisateur fait le choix de minimiser le diamètre, cela revient à minimiser D. Cependant, le nombre de contraintes réifiées est quadratique par rapport au nombre d'objets et la gestion des contraintes réifiées nécessite des variables et des contraintes supplémentaires. Avec la minimisation de D par *branch-and-bound*, nous remplaçons ces contraintes par :

- A chaque solution intermédiaire Δ, calculer le diamètre $D(\Delta)$.
- Pour chaque paire $o < o' \in \mathcal{O}$ telle que $d(o, o') > D(\Delta)$, poser la contrainte $G[o] \neq G[o']$.

Maximiser la marge La marge minimale entre clusters est représentée par une variable S du domaine des nombres flottants. De la même façon que pour minimiser le diamètre, lorsque l'utilisateur souhaite maximiser la marge entre clusters, la valeur de la variable S est maximisée. Puisque S représente la marge minimale entre clusters, deux objets $o, o' \in \mathcal{O}$ vérifiant $d(o, o') < S$ doivent être dans le même cluster. La minimisation de S se réalise par :

- A chaque solution intermédiaire Δ, calculer la marge $S(\Delta)$.
- Pour chaque paire $o < o' \in \mathcal{O}$ telle que $d(o, o') < S(\Delta)$, poser la contrainte $G[o] = G[o']$.

Maximiser le nombre d'objets par cluster Si l'utilisateur choisit de maximiser le nombre d'objets par cluster, une variable F qui représente la fréquence des clusters est introduite. Son domaine est l'ensemble $[1, n]$. Pour chaque valeur $c \in \mathcal{C}$, nous posons une contrainte *atleast*(F, G, c) pour indiquer que F est la fréquence du cluster c. La valeur de F est maximisée.

Maximiser le nombre d'attributs caractérisant les clusters Dans ce cas, une variable T du domaine $[1, m]$ est introduite. Les contraintes suivantes permettent d'associer T au nombre minimal d'attributs des clusters, dans le cas du modèle avec des variables binaires A :

$$\forall c \in \mathcal{C},\ T \leq \sum_{a \in \mathcal{A}} A[c, a]$$

ou dans le cas du modèle avec des variables ensemblistes E :

$$\forall c \in \mathcal{C},\ T \leq |E[c]|$$

La valeur de T est maximisée.

4 Expérimentations

Les modèles présentés précédemment ont été implantés en utilisant la version 4.2.1 de la bibliothèque de programmation par contrainte Gecode [2]. Les expérimentations suivantes ont deux objectifs : comparer l'efficacité des modèles binaire et ensembliste et montrer l'intérêt de notre cadre unifié.

Nous testons notre approche sur la base *Automobile* disponible dans le répertoire UCI [3]. Après retrait des données incomplètes, la base contient 193 objets (des modèles d'automobiles) et 23 attributs. 22 attributs (9 symboliques et 13 réels) concernent des caractéristiques techniques des véhicules. Par exemple, une automobile est caractérisée par un type de motorisation (diesel ou essence), des roues motrices (4 roues motrices, 2 à l'avant ou 2 à l'arrière), par une puissance (comprise entre 48 to 288),... Le dernier attribut est le prix (ceux-ci vont de 5118 à 45400).

Nous choisissons de prendre ce prix pour calculer la dissimilarité entre les véhicules, la distance entre chaque véhicule étant la différence de prix. Cela permet d'obtenir des résultats aisément interprétables. Les caractéristiques techniques sont quant à elles discrétisées et fournissent 64 attributs pour le côté conceptuel (pour chaque donnée réelle, deux attributs sont créés pour répartir les objets selon leur position par rapport à la médiane).

Afin d'exploiter le partitionnement des objets, le branchement est d'abord effectué sur les objets (G) en les prenant par ordre croissant, puis sur les attributs (A ou E selon le modèle).

4.1 Comparaison des modèles binaire et ensembliste

La différence entre les modèles binaire et ensembliste porte sur les contraintes conceptuelles. Les tests sont

2. `http://www.gecode.org`
3. `http://archive.ics.uci.edu/ml`

Contr.	k	Modèle bin.		Modèle ens.	
		time(s)	#nodes	time(s)	#nodes
int.	2	0.10	2338	0.14	2877
	3	0.63	15109	0.55	13986
ext.	2	0.03	79	0.03	52
	3	8.40	145098	0.08	979
	4	6960	>1M	0.55	8636
	5	_	_	2.49	36264
ferm.	2	0.03	24	0.03	47
	3	13.50	72494	0.15	963
	4	9900	>1M	1.29	8585
	5	_	_	5.47	36057

TABLE 1 – Approches binaire et ensembliste

Test	Contrainte	Optimisation	#cl.
(a)	fermeture	#obj./cl.	2
(b)	fermeture	#att./cl.	2
(c)	intention	diamètre	2
(d)	fermeture	diamètre	2
(e)	fermeture	#obj./cl.	3
(f)	fermeture	#att./cl	3
(g)	intention	diamètre	3
(h)	fermeture	diamètre	3

TABLE 2 – Tests

donc effectués respectivement avec la contrainte d'intention, la contrainte d'extension et enfin la conjonction des deux (i.e. la fermeture). L'optimisation est réalisée en minimisant le diamètre du plus grand cluster. Les performances obtenues sont résumées dans la table 1 (le temps de calcul est limité à 3h, le caractère _ indique que ce temps de calcul est dépassé).

Les résultats mettent clairement en évidence l'efficacité du modèle ensembliste par rapport au modèle binaire. Quel que soit le modèle choisi, la contrainte d'intention seule ne permet pas de réduire suffisamment l'espace de recherche et se révèle peu efficace. La différence entre les deux modèles porte sur la contrainte d'extension (que l'on retrouve également dans la fermeture). L'utilisation de contraintes réifiées liées à des contraintes linéaires dans le cas binaire est très pénalisante. Au contraire, les contraintes d'inclusion du modèle ensembliste permettent une bonne propagation et un nombre de nœuds à explorer beaucoup moins important.

4.2 Comparaison avec des approches conceptuelles ou relationnelles

Les tests suivants sont réalisés avec le modèle ensembliste. L'objectif est d'obtenir des concepts suivant le prix avec 2 ou 3 clusters afin que les résultats obtenus soient les plus interprétables possible. Plusieurs types de tests (listés dans la table 2) sont réalisés : avec une approche purement conceptuelle, avec une approche purement relationnelle et enfin en utilisant notre cadre unifié.

Expérimentations avec 2 clusters Les premières expérimentations sont réalisées avec 2 clusters, les tailles des clusters obtenus sont données dans la table 3 et la figure 1 indique la répartition des objets dans les clusters en fonction de leur prix.

	Clusters				
test	#obj./cl.		#att./cl.		diam.
(a)	95	98	2	2	40282
(b)	94	99	2	2	39901
(c)	176	17	1	7	19848
(d)	98	95	1	2	37387

TABLE 3 – Taille des clusters pour k=2

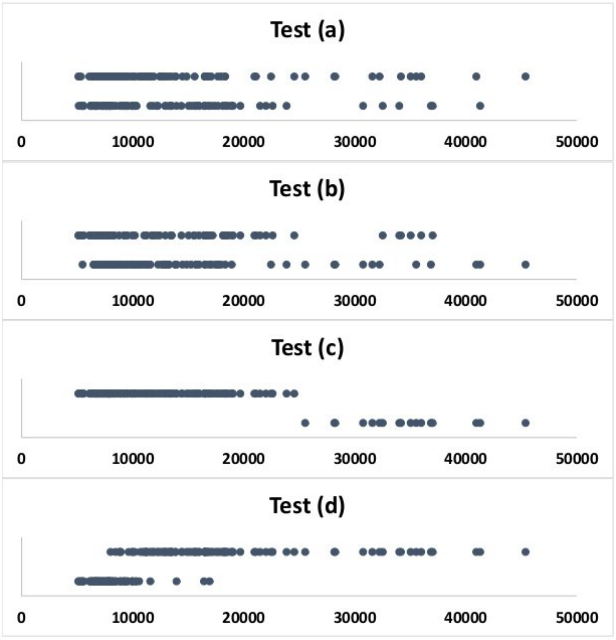

FIGURE 1 – Répartition des objets/cluster pour k=2

Cadre conceptuel : les tests (a) et (b) sont réalisés avec la contrainte de fermeture en maximisant respectivement la taille des clusters (a) et la taille des concepts (b). La distance n'intervenant pas, on se place donc ici dans un cadre purement conceptuel. Pour les deux tests, la taille des clusters est équilibrée. Alors qu'on aurait pu espérer que les prix soient corrélés aux caractéristiques techniques, les clusters obtenus ne représentent pas de regroupement de véhicules par prix. L'intégration des prix dans les attributs conceptuels n'est pas présentée ici car elle ne modifie pas les clusters obtenus.

Cadre relationnel : le test (c) est réalisé en minimisant le diamètre et en utilisant la contrainte d'intention pour obtenir les attributs communs aux objets de chaque cluster. En se plaçant ainsi dans un cadre purement relationnel, le diamètre est forcément optimal et les intervalles de prix des clusters totalement disjoints. On obtient une bonne caractérisation du second cluster par 7 attributs (roues motrices à l'arrière, poids élevé, moteur puissant, forte consommation...) du fait de sa petite taille. Au contraire, le premier cluster n'est lui caractérisé que par 1 attribut (moteur à l'avant). La différence de taille des clusters s'explique par des différences de prix très importantes dans le haut de gamme. Notez qu'utiliser l'ensemble des attributs dans le calcul de dissimilarité ne permet pas d'équilibrer davantage la taille des clusters.

Cadre unifié : en remplaçant la contrainte d'intention par la contrainte de fermeture, le test (d) permet d'obtenir des concepts tout en minimisant le diamètre. La taille des clusters obtenus est équilibrée même si les concepts contiennent peu d'attributs (moteur à l'avant et poids faible pour l'un, poids élevé pour l'autre), ce qui est logique étant donné le nombre de clusters et l'utilisation de la contrainte de fermeture. La répartition des objets fait apparaître que le premier cluster correspond aux véhicules à bas coût et le second aux véhicules plus onéreux. La différence de diamètre entre les deux clusters s'explique, comme précédemment par des écarts de prix plus importants dans les modèles haut de gamme. L'extension permet ici de mettre en évidence le poids comme attribut discriminant.

Expérimentations avec 3 clusters Passer de deux à trois clusters permet de réduire leur taille et par conséquent d'augmenter la taille des concepts. Les résultats des expérimentations pour 3 clusters sont résumés dans la table 4 et la figure 2.

Cadre conceptuel : les tests (e) et (f) sont réalisés avec la contrainte de fermeture en maximisant respectivement le nombre d'objets et le nombre d'attributs (on se place donc à nouveau dans un cadre purement conceptuel). Les résultats obtenus par les 2 méthodes

	Clusters						
test	#objets			#attributs			diam.
(e)	63	66	64	3	3	3	38615
(f)	68	61	64	3	3	3	38615
(g)	161	20	12	1	5	10	13226
(h)	72	108	13	4	2	5	33550

TABLE 4 – Taille des clusters pour k=3

sont très proches, que ce soit au niveau de la taille des clusters et des concepts, qu'au niveau du contenu de ces concepts (seul un attribut varie : le poids du véhicule est remplacé par sa hauteur). Cependant dans ces deux cas encore, comme pour les tests sur deux clusters, la répartition selon les prix est peu probante.

Cadre relationnel : le test (g) réalisé comme pour (c) dans un cadre purement relationnel fournit toujours un diamètre optimal et des objets parfaitement répartis par prix. Cependant, la taille des clusters est toujours très déséquilibrée.

Cadre unifié : le test (h) permet d'obtenir des clusters de tailles plus équilibrées que l'approche relationnelle et reflétant davantage les gammes d'automobile que l'approche conceptuelle. Le premier cluster caractérise 72 véhicules par 4 attributs : un poids élevé, un moteur de grande taille, une puissance élevée et une consommation sur autoroute importante. Le second cluster contient 108 véhicules caractérisés par 2 attributs : un moteur à l'avant et une consommation sur autoroute faible. Pour le dernier cluster, les 13 véhicules sont caractérisés par 5 attributs : un moteur essence, situé à l'avant, de petite taille, avec 4 cylindres et une consommation sur autoroute élevée. Le premier cluster correspond clairement aux véhicules haut de gamme alors que les deux derniers clusters contiennent des véhicules de prix plus bas.

Intérêt du cadre unifié Les résultats obtenus précédemment démontrent l'intérêt d'une méthode unifiée qui tire parti des points forts des approches conceptuelle et relationnelle. Les tests (i) et (j) montrent comment améliorer encore les résultats grâce à l'apport des contraintes utilisateur et à une exploitation plus poussée de la dissimilarité. Les résultats obtenus se trouvent dans la table 5 et la figure 3.

Le test (i) est réalisé dans le cas d'une recherche de 3 clusters en optimisant le diamètre, avec la contrainte de fermeture et en ajoutant deux contraintes utilisateur : une taille de clusters supérieure ou égale à 40 et une taille de concepts supérieure ou égale à 2. Par rapport au test (h) réalisé sans ces contraintes utilisateur, il permet, moyennant une augmentation des diamètres

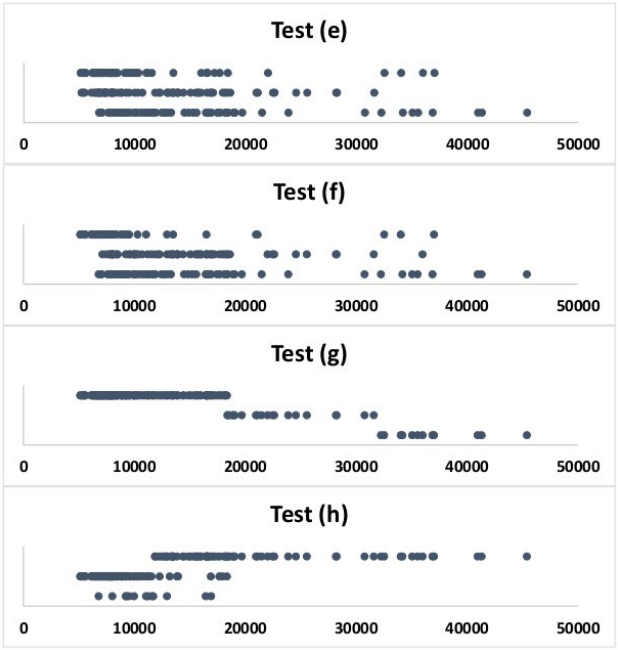

FIGURE 2 – Répartition des objets/cluster pour k=3

des clusters, d'obtenir des tailles de clusters plus équilibrées. Le premier cluster caractérise les véhicules avec un poids et un ratio de compression important. Les véhicules du second cluster sont caractérisés par un poids faible et un moteur à l'avant. Enfin les caractéristiques du dernier cluster sont : un moteur essence, situé à l'avant, un poids important et un ratio de compression faible.

Enfin, notre cadre unifié n'est pas restreint à des dissimilarités calculées sur un seul attribut. Pour notre exemple, on peut considérer qu'un véhicule haut de gamme est à la fois cher et puissant. Le test (j) est ainsi réalisé en calculant une dissimilarité prenant en compte à la fois la puissance et le prix du véhicule. En utilisant les mêmes contraintes que précédemment, les résultats obtenus font apparaître des objets mieux répartis dans les clusters et des concepts bien définis. Le premier cluster caractérise les véhicules haut de gamme par les attributs essence, taille de moteur élevée, système d'injection mpfi, course du piston faible, forte consommation urbaine et sur autoroute. Les véhicules du second cluster sont caractérisés par un moteur à l'avant et de grande taille ainsi qu'une course du piston élevée. Enfin, le dernier cluster regroupe des véhicules de bas de gamme caractérisés par un moteur positionné à l'avant et de petite taille.

test	Clusters						
	#objets			#attributs			diam.
(i)	57	95	41	2	2	4	36479
(j)	42	56	95	7	3	2	38411

TABLE 5 – Taille des clusters pour (i) et (j)

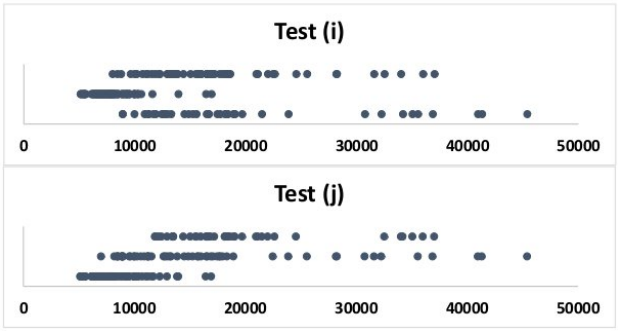

FIGURE 3 – Répartition des objets/cluster pour les tests (i) et (j)

5 Conclusion

Dans cet article, nous avons proposé un cadre à base de programmation par contraintes, qui unifie le clustering relationnel et le clustering conceptuel. Ce cadre complet offre différentes possibilités, car il peut se décliner en des tâches de clustering purement conceptuel ou purement relationnel, ou permettre de réaliser des tâches de clustering conceptuel (ou relationnel) en intégrant à la fois des contraintes de structure, relationnelles ou conceptuelles. Nous avons présenté deux réalisations pour les contraintes conceptuelles : une réalisation binaire utilisant des variables et contraintes booléennes et une réalisation utilisant des variables et contraintes ensemblistes. Enfin, des exemples concrets ont démontré l'intérêt de ce cadre unifié pour l'utilisateur.

Nous continuons à développer ce cadre général dans plusieurs directions. Le choix du nombre k de clusters est pour l'instant une contrainte dure du modèle. Le passage de k en variable du problème implique l'utilisation d'une fonction objective plus complexe que celles proposées ici. En effet, minimiser le diamètre, maximiser la marge ou maximiser la taille des concepts pousse k vers des valeurs élevées. Au contraire, maximiser la taille des clusters tend à amener k vers des valeurs faibles. Nous souhaitons également améliorer l'efficacité de notre cadre pour pouvoir traiter des bases de données de plus grandes dimensions. Enfin, nous souhaitons étendre le modèle par l'intégration de nouvelles contraintes conceptuelles.

Références

[1] Behrouz Babaki, Tias Guns, and Siegfried Nijssen. Constrained clustering using column generation. In *CPAIOR*, pages 438–454, 2014.

[2] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. A Declarative Framework for Constrained Clustering. In *ECMLPKDD*, pages 419–434, 2013.

[3] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Clustering sous contraintes en PPC. *Revue d'Intelligence Artificielle*, 28(5) :523–545, 2014.

[4] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Constrained clustering by constraint programming. *Artificial Intelligence*, To appear.

[5] Ian Davidson and S. S. Ravi. Clustering with Constraints : Feasibility Issues and the k-Means Algorithm. In *ICDM*, pages 138–149, 2005.

[6] Ian Davidson and S. S. Ravi. The Complexity of Non-hierarchical Clustering with Instance and Cluster Level Constraints. *Data Min. Knowl. Disc.*, 14(1) :25–61, 2007.

[7] Ian Davidson, S. S. Ravi, and Leonid Shamis. A SAT-based Framework for Efficient Constrained Clustering. In *ICDM*, pages 94–105, 2010.

[8] Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint programming for itemset mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 204–212, 2008.

[9] Sean Gilpin, Siegfried Nijssen, and Ian N. Davidson. Formalizing hierarchical clustering as integer linear programming. In *AAAI*, pages 372–378, 2013.

[10] Tias Guns, Siegfried Nijssen, and Luc De Raedt. Itemset mining : A constraint programming perspective. *Artificial Intelligence*, 175 :1951–1983, 2011.

[11] Tias Guns, Siegfried Nijssen, and Luc De Raedt. k-Pattern set mining under constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(2) :402–418, 2013.

[12] Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. The top-k frequent closed itemset mining using top-k SAT problem. In *ECMLPKDD*, pages 403–418, 2013.

[13] Matti Järvisalo. Itemset mining as a challenge application for answer set enumeration. In *LPNMR*, pages 304–310, 2011.

[14] Yat Chiu Law and Jimmy Ho-Man Lee. Global constraints for integer and set value precedence. In Mark Wallace, editor, *CP*, pages 362–376, 2004.

[15] Jean-Philippe Métivier, Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, and Samir Loudni. Constrained Clustering Using SAT. In *IDA*, pages 207–218, 2012.

[16] Marianne Mueller and Stefan Kramer. Integer Linear Programming Models for Constrained Clustering. In *DS*, pages 159–173, 2010.

[17] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *ICML*, pages 1103–1110, 2000.

[18] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained K-means Clustering with Background Knowledge. In *ICML*, pages 577–584, 2001.

[19] Xiang Wang, Buyue Qian, and Ian Davidson. On constrained spectral clustering and its applications. *Data Min. Knowl. Discov.*, 28(1) :1–30, 2014.