Un modèle général pour la classification non supervisée sous contraintes d'utilisateur

Thi-Bich-Hanh Dao, Khanh-Chuong Duong, Christel Vrain

LIFO, Université d'Orléans, 45067 Orléans cedex 02, France

{thi-bich-hanh.dao, khanh-chuong.duong, christel.vrain}@univ-orleans.fr

Résumé

La classification non supervisée est une tâche importante dans le domaine de la Fouille de Données et depuis le début des années 2000, on tend à intégrer dans ce processus des contraintes afin de mieux modéliser les attentes des utilisateurs. Cependant, les systèmes développés sont en général adaptés à certains types de contraintes seulement. Nous proposons un modèle général pour la classification non supervisée sous contraintes en Programmation par Contraintes. Il permet d'intégrer des contraintes, qu'elles portent sur des instances ou sur les groupes recherchés. Différents critères d'optimisations sont intégrés au modèle : la minimisation du diamètre maximal, la maximisation de la marge entre les clusters ou la minimisation de la somme des dissimilarités intracluster. Des expériences sur des ensembles de données classiques montrent l'intérêt de notre approche.

Abstract

Cluster analysis is an important task in Data Mining and since the early 2000s user-defined constraints have been integrated in order to better model real-world applications. Well-known algorithms are extended to handle user-constraints, however usually for one kind of constraints. We present a general model for constrained clustering in Constraint Programming, which allows to integrate directly different kinds of user-constraints. Different criteria are included: minimizing the maximal diameter, maximizing the separation between clusters or minimizing the within-cluster sum of dissimilarities. Experiments on classical datasets show the interest of our approach.

1 Introduction

La problématique de classification non supervisée (aussi appelée clustering) a déjà été longuement étudiée. C'est un champ de recherche difficile pour plusieurs raisons : le choix de la mesure de dissimilarité entre les objets dépendant principalement de l'application mais influant fortement sur les résultats, la définition du critère à optimiser, la taille de l'espace de recherche avec pour conséquence la nécessité de définir des heuristiques conduisant souvent à un optimum local. Poser des contraintes sur la solution recherchée permet d'une part, de modéliser plus finement les applications réelles et d'autre part de restreindre la taille de l'espace de recherche. Néanmoins, la plupart des algorithmes classiques n'ont pas été développés pour la classification non supervisée sous contraintes et doivent être adaptés, si possible, pour prendre en compte les contraintes posées par l'utilisateur. Développer des solveurs généraux applicables à une grande variété de problèmes pose de nouveaux défis.

D'autre part, des avancées récentes en Programmation par Contraintes (PPC) ont rendu ce paradigme beaucoup plus puissant. Plusieurs travaux [10, 9, 4] ont étudié l'intérêt de la PPC pour modéliser des problèmes de fouille de données et ont montré l'apport de la déclarativité inhérente à la PPC.

Dans des travaux récents [5], nous avons proposé une approche en PPC pour la classification non supervisée sous contraintes d'utilisateur avec comme critère la minimisation du diamètre maximal des classes. L'intérêt de notre approche est de fournir un modèle déclaratif permettant de spécifier le problème de classification non supervisée et d'intégrer facilement des contraintes d'utilisateur. Dans ce papier, nous généralisons le modèle avec plusieurs critères d'optimisation : minimiser le diamètre maximal, maximiser la marge entre les classes ou minimiser la somme des dissimilarités intraclasse. Dans notre modèle, nous faisons l'hypothèse que nous disposons d'une mesure de dissimilarité entre les paires d'objets et que le nombre k de classes est fixé (en théorie aucune limite n'est donnée sur la valeur de k, mais plus k est grand, plus la complexité est élevée).

Nous montrons que notre cadre intègre naturellement des contraintes d'utilisateur connues, par exemple des contraintes sur les instances (must-link, cannot-link) ou des contraintes sur les classes (capacité maximale ou minimale, séparation, diamètre). Nous avons également généralisé une contrainte sur les classes, permettant de modéliser une notion de densité.

Il est reconnu en PPC que le choix de variables et de contraintes est fondamental, mais que la stratégie de recherche est tout aussi importante pour améliorer l'efficacité. Des améliorations du modèle sont étudiées, reposant sur des résultats théoriques ce qui permet d'alléger le modèle sans changer la sémantique. De plus, parmi les critères intégrés, celui de la somme des dissimilarités intraclasse se représente par une somme linéaire. Afin d'avoir une propagation plus efficace pour ce critère, nous avons implanté un algorithme de filtrage qui exploite des connaissances lors d'une affectation partielle de variables. Des expérimentations sur des bases de données classiques montrent que notre approche est capable de traiter ces bases en un temps raisonnable et surtout, que la possibilité de combiner différents types de contraintes d'utilisateur permet d'obtenir des solutions intéressantes. Contrairement à la plupart des travaux existants dans la classification non supervisée sous contraintes d'utilisateur, notre modèle permet de trouver un optimum global.

Des travaux récents [14, 16] ont déjà proposé d'utiliser la PPC pour la classification non supervisée conceptuelle dans des bases de données transactionnelles. Le problème est formalisé comme la recherche d'un ensemble de k-motifs fréquents, deux à deux non recouvrants, dont l'ensemble couvre toutes les données : les transactions sont vues comme des objets et les motifs comme des définitions en intension des classes. Plusieurs critères d'optimisation sont considérés comme maximiser la taille minimale d'une classe ou minimiser la différence entre les tailles des classes. Une approche de modélisation par programmation linéaire sur des entiers a été proposée dans [17]. Dans cette approche un ensemble de clusters candidat doit être connu à l'avance, et le modèle permet de trouver un clustering formé par le meilleur sous-ensemble parmi les candidats. Cette approche est expérimentée pour la classification non supervisée conceptuelle, les clusters candidat sont des motifs fréquents. Notons que ces approches sont donc adaptées à des bases de données qualitatives alors que notre approche peut traiter tout type de données dès lors que l'on dispose d'une mesure de dissimilarité entre les données. Un cadre SAT pour la classification non supervisée sous contraintes est proposé en [8], mais uniquement pour un problème à 2 classes ($k = 2$) : il traite les contraintes sur les instances (must-link et cannot-link) et des contraintes sur les classes (diamètre des clusters, séparation entre les clusters). Son algorithme converge vers un optimum global. Notre approche est plus générale dans la mesure où différents autres critères sont traités et le nombre de classes n'est pas limité à 2.

Le papier est organisé comme suit. Dans la section 2, nous rappelons des notions sur la classification non supervisée et celle sous contraintes d'utilisateur. La section 3 est dédiée à la présentation de notre modèle et la section 4 aux expérimentations. La conclusion et une discussion sur les travaux futurs sont données dans la section 5.

2 Préliminaires

2.1 Classification non supervisée

La classification non supervisée (ou clustering) consiste à regrouper les données dans des classes (ou clusters), de manière à regrouper dans un même cluster les données similaires et à séparer les données distantes dans des clusters différents. Etant donnés une base de données de n points (objets) $\mathcal{O} = \{o_1, \ldots, o_n\}$ d'un espace $\mathcal{X}$ et une mesure de dissimilarité $d(o_i, o_j)$ entre deux points o_i et o_j, l'objectif est de regrouper les points en différentes classes de telle manière que la partition obtenue optimise un critère donné. Le problème de clustering peut être donc formulé comme un problème d'optimisation.

Les problèmes de clustering sont variés, dépendant de différents critères, comme la structure souhaitée (une partition, une hiérarchie, des classes recouvrantes, ...), ou encore le critère à optimiser. Dans ce papier, nous nous intéressons à la recherche d'une partition des points en k classes $C_1, \ldots, C_k$ telle que : (1) pour tout $c \in [1, k]$, $C_c \neq \emptyset$, (2) $\cup_c C_c = \mathcal{O}$, (3) pour tout $c \neq c'$, $C_c \cap C_{c'} = \emptyset$, et (4) un critère E est optimisé. Le critère optimisé peut être, entre autres :

- Minimisation des moindres carrés : $E = \sum_{c=1}^{k} \sum_{o_i \in C_c} d(m_c, o_i)^2$, où m_c est le centre de chaque cluster C_c.
- Minimisation de la somme de dissimilarités intra-cluster : $E = \sum_{c=1}^{k} \sum_{o_i, o_j \in C_c} d(o_i, o_j)^2$. Ce critère, standardisé en divisant par la taille de chaque cluster, est équivalent au critère des moindres carrés dans le cas de la distance euclidienne.
- Minimisation de l'erreur absolue : $E = \sum_{c=1}^{k} \sum_{o_i \in C_c} d(o_i, r_c)$, où r_c est le représentant du cluster C_c.
- Minimisation du diamètre maximal : $E = max_{c \in [1,k], o_i, o_j \in C_c}(d(o_i, o_j))$. E est le diamètre maximal des clusters, où le diamètre d'un cluster est la distance maximale entre chaque paire

de ses points.
- Maximisation du séparateur minimal : $E = min_{c<c' \in [1,k], o_i \in C_c, o_j \in C_{c'}}(d(o_i, o_j))$. E est le séparateur minimal des clusters, où le séparateur de deux clusters $C_c, C_{c'}$ est la valeur minimale des distances $d(o_i, o_j)$, avec $o_i \in C_c$ et $o_j \in C_{c'}$.

L'algorithme k-means représente chaque cluster par la moyenne des points du cluster et tend à minimiser le critère des moindres carrés, alors que l'algorithme k-médoïdes choisit parmi les points un représentant de chaque cluster et cherche à minimiser le critère d'erreur absolue. A chaque itération, les deux méthodes réduisent la valeur du critère jusqu'à un optimum, en général local.

L'algorithme FPF (Furthest Point First) [13] minimise le diamètre maximal des clusters. L'algorithme commence par choisir un point comme le représentant du premier cluster et affecte tous les points à ce cluster. A l'itération suivante, le point le plus loin du premier représentant est choisi comme représentant du second cluster. Les points qui sont plus proches du second représentant que du premier sont réaffectés au second cluster. L'algorithme réitère les 2 étapes : choisir comme nouveau représentant le point le plus loin des représentants existants et réaffecter les points. Il s'arrête après k itérations, ayant ainsi formé k clusters. La complexité en temps est $O(nk)$. Si d_{opt} est l'optimum global alors l'algorithme garantit de trouver un clustering dont le diamètre maximal des clusters d vérifie $d_{opt} \leq d \leq 2d_{opt}$, si la mesure utilisée satisfait l'inégalité triangulaire. De plus, dans [13] il a été prouvé que trouver d tel que $d \leq (2-\epsilon)d_{opt}$ est NP-Difficile pour tout $\epsilon > 0$, lorsque les points sont dans un espace à 3 dimensions. L'algorithme modifié (M-FPF) proposé dans [12] trouve une solution identique à FPF tout en étant plus rapide en temps.

La plupart des méthodes de partitionnement trouvent des groupes sous forme sphérique. Des approches, comme par exemple DBSCAN [11] basée sur la notion de densité et ne reposant sur aucun critère d'optimisation, permettent de découvrir des groupes de forme arbitraire. DBSCAN nécessite deux paramètres d'entrée : une distance ϵ et un nombre minimum de points $MinPts$. Un point est un noyau si son voisinage de rayon ϵ contient au moins $MinPts$ points. La méthode considère un à un tous les points non affectés à un cluster, si le point est un noyau, un nouveau cluster est créé et les points dans son voisinage y sont ajoutés. L'algorithme étend ensuite le cluster de façon récursive en y ajoutant tout le voisinage des points noyaux du cluster. Cette méthode se base donc seulement sur des paramètres de densité ϵ et $MinPts$, il n'y a pas de critère à optimiser et le nombre de classes k n'est pas fixé à l'avance. Dans OPTICS [1], un algorithme qui peut être vu comme une généralisation de DBSCAN, la valeur de ϵ est remplacée par une borne maximale. Cet algorithme permet d'identifier ϵ lorsque le nombre k est fixé.

2.2 Classification non supervisée sous contraintes d'utilisateurs

Les contraintes définies par l'utilisateur sur la solution recherchée permettent de modéliser plus finement des applications réelles. Ce sont des contraintes portant sur des instances (des points) ou sur des clusters.

Les contraintes portant sur des instances les plus utilisées sont les contraintes "must-link" et "cannot-link", introduites pour la première fois dans [21] . Ces contraintes sont posées sur des paires d'objets individuels. Une contrainte must-link indique que deux points o_i et o_j doivent être dans le même cluster :

$$\forall c \in [1,k],\ o_i \in C_c \Leftrightarrow o_j \in C_c$$

Une contrainte cannot-link indique que deux points ne peuvent pas appartenir au même cluster :

$$\forall c \in [1,k],\ \neg(o_i \in C_c \wedge o_j \in C_c)$$

Les contraintes portant sur des clusters imposent des conditions sur la forme, la taille ou d'autres caractéristiques. Une contrainte sur la taille peut être une contrainte de capacité minimale, qui indique que chaque cluster doit avoir au moins α points, permettant ainsi d'éviter des clusters trop petits :

$$\forall c \in [1,k],\ |C_c| \geq \alpha,$$

ou encore une contrainte de capacité maximale, qui indique que chaque cluster doit avoir au plus β points :

$$\forall c \in [1,k],\ |C_c| \leq \beta.$$

Comme contrainte sur la forme, l'utilisateur peut limiter le diamètre des clusters :

$$\forall c \in [1,k], \forall o_i, o_j \in C_c,\ d(o_i, o_j) \leq \gamma.$$

Il existe d'autres types de contraintes sur des clusters comme par exemple la δ-contrainte et la ϵ-contrainte [7]. Une δ-contrainte ou contrainte de séparation impose que chaque paire de points appartenant à deux groupes différents soit à une distance d'au moins δ :

$$\forall c < c' \in [1,k], \forall o_i \in C_c, o_j \in C_{c'}, d(o_i, o_j) \geq \delta.$$

Une ϵ-contrainte impose que chaque point o_i ait dans un rayon ϵ un autre point du même cluster :

$$\forall c \in [1,k], \forall o_i \in C_c, \exists o_j \in C_c, o_j \neq o_i \text{ et } d(o_i, o_j) \leq \epsilon.$$

Cette contrainte est basée sur la notion de densité du DBSCAN. Dans la même idée que DBSCAN, nous proposons une nouvelle contrainte de densité plus générale que ϵ-contrainte, qui impose que chaque point o_i doit avoir dans un rayon de ϵ au moins $MinPts$ points qui sont dans le même cluster que o_i.

Depuis une dizaine d'années, différents travaux ont développé des extensions des algorithmes classiques aux contraintes must-link et cannot-link, comme par exemple COBWEB [21], k-means [22, 3], classification non supervisée hierarchique [6] ou spectrale [15, 23], etc. L'extension est faite en modifiant la mesure de dissimilarité, la fonction objectif ou la stratégie de recherche. Cependant à notre connaissance il n'existe pas de solution générale pour étendre un algorithme traditionnel en y intégrant simultanément différents types de contraintes.

3 Modélisation des problèmes de clustering sous contraintes

Nous disposons d'une collection de n points et d'une mesure de dissimilarité entre ces points. Sans perte de généralité nous supposons que les points sont indexés et nommés par leur indice. Nous considérons le cas où le nombre de clusters k est connu à l'avance.

3.1 Modèle

Variables Nous fixons pour chaque cluster un représentant, qui est le point de plus petit indice de ce cluster. Nous introduisons donc une variable entière $I[c]$ pour chaque $c \in [1, k]$, dont la valeur est le représentant du cluster c. Le domaine de chaque $I[c]$ est $[1, n]$, car tout point peut être représentant d'un cluster.

Affecter un point à un cluster devient associer ce point au représentant du cluster. Nous introduisons pour chaque point $i \in [1, n]$ une variable entière $G[i]$ du domain $[1, n]$ qui donne le représentant associé.

Nous introduisons également une variable qui représente la valeur du critère à optimiser (D s'il s'agit du diamètre maximal, S la séparation minimale, V la somme des dissimilarités intra-cluster). C'est une variable entière, car nous utilisons des solveurs sur des variables entières. Le domaine de D et de S est l'intervalle formé par la distance minimale et la distance maximale entre chaque paire de points. Le domaine de V est borné supérieurement par la somme des distances au carré de toutes les paires de variables. Dans nos expérimentations, lorsque la distance n'est pas entière, elle est multipliée par 100 et seule la partie entière est conservée.

Modélisation d'une partition La relation entre les points et leur cluster est exprimée par ces contraintes :

- Le représentant d'un représentant est lui-même :

$$\forall c \in [1, k],\ G[I[c]] = I[c].$$

- Le représentant d'un point doit être parmi les représentants : la valeur de chaque $G[i]$ doit être présente une fois dans I

$$\forall i \in [1, n],\quad \#\{c \mid I[c]{=}G[i]\} = 1.$$

- Le représentant doit être d'indice minimal, l'indice de chaque point est donc supérieur ou égale à celui du représentant : $\forall i \in [1, n],\quad G[i] \leq i$.

Un même ensemble de clusters pourrait être représenté de différentes manières, selon l'ordre des clusters. Les contraintes suivantes permettent d'éviter cette symétrie :

- Les représentants sont en ordre croissant : $\forall c < c' \in [1, k],\ I[c] < I[c']$.
- Le représentant du premier cluster est le premier point : $I[1] = 1$.

Modélisation de différents critères d'objectif S'il s'agit de minimiser le diamètre maximal des clusters, les contraintes suivantes sont ajoutées :

- Deux points à une distance supérieure au diamètre maximal doivent être dans des clusters différents. Ceci est représenté par les contraintes réifiées suivantes : $\forall i < j \in [1, n]$,

$$d(i, j) > D \rightarrow G[j] \neq G[i] \tag{1}$$

- Le diamètre maximal est minimisé : minimise D.

S'il s'agit de maximiser la séparation minimale entre clusters :

- Deux points à une distance inférieure à la séparation minimale doivent être dans le même cluster. Ceci est représenté par les contraintes réifiées suivantes : $\forall i < j \in [1, n]$,

$$d(i, j) < S \rightarrow G[j] = G[i] \tag{2}$$

- La séparation est maximisée : maximise S.

Si le critère est de minimiser la somme des dissimilarités, minimise V avec :

$$V = \sum_{i,j \in [1,n]} (G[i] == G[j]) d(i, j)^2 \tag{3}$$

Modélisation des contraintes définies par l'utilisateur Plusieurs contraintes définies par l'utilisateur, sur des clusters ou sur des points, peuvent s'ajouter directement au modèle. Pour les contraintes portant sur les clusters :

– La capacité minimale α des clusters : $\forall c \in [1,k]$,

$$\#\{i \mid G[i]{=}I[c]\} \geq \alpha$$

– La capacité maximale β des clusters : $\forall c \in [1,k]$,

$$\#\{i \mid G[i]{=}I[c]\} \leq \beta$$

– La δ-contrainte indique que la séparation entre deux clusters doit être au moins δ, ainsi deux points à une distance inférieure à δ doivent être dans le même cluster. Pour chaque $i < j \in [1,n]$ tels que $d(i,j) < \delta$, nous posons la contrainte : $G[i] = G[j]$.
– La contrainte de diamètre indique que le diamètre de chaque cluster doit être au plus γ, ainsi deux points à une distance supérieure à γ doivent être dans des clusters différents. Pour chaque $i < j \in [1,n]$ tels que $d(i,j) > \gamma$, nous posons : $G[j] \neq G[i]$.
– La contrainte de densité indique que chaque point i doit avoir dans un rayon de ϵ, au moins $MinPts$ points dans le même cluster que i. Donc, pour chaque $i \in [1,n]$, l'ensemble de points qui sont dans un rayon de ϵ est calculé, sur lequel la contrainte suivante est posée :

$$\#\{j \mid d(i,j) \leq \epsilon, G[j]{=}G[i]\} \geq MinPts$$

Pour les contraintes portant sur les couples de points :
– Une contrainte must-link sur i,j se traduit par $G[i] = G[j]$.
– Une contrainte cannot-link : $G[i] \neq G[j]$.

L'ajout de ces contraintes implique également des contraintes sur D ou S, par exemple $G[i] = G[j]$ implique $D \geq d(i,j)$ et $G[i] \neq G[j]$ implique $S \leq d(i,j)$.

Stratégie de recherche Les variables sont choisies dans l'ordre des variables de I puis celles de G. Cet ordre indique que les représentants de cluster doivent être identifiés en premier, puis le solveur essaye de déterminer l'affectation de points aux clusters.

Les variables de I sont choisies de $I[1]$ à $I[k]$, c'est-à-dire du premier au dernier représentant. Puisqu'un représentant doit avoir l'indice minimal dans le cluster, les valeurs pour chaque $I[c]$ sont choisies dans l'ordre croissant.

Les variables de G sont choisies dans l'ordre croissant sur la taille du domaine restant. Pour le choix de valeur pour chaque $G[i]$, l'indice du représentant le plus proche est choisi en premier.

3.2 Améliorations du modèle

Les solveurs de PPC réalisant une recherche exhaustive, ce modèle permet de trouver la solution optimale. Afin d'améliorer l'efficacité du modèle, différents aspects sont considérés.

Amélioration de la recherche en réordonnant les points Afin d'identifier les représentants, les variables de I sont choisies dans l'ordre $I[1], \ldots, I[k]$. Puisqu'un représentant doit avoir l'indice minimal, pour chaque variable, les valeurs (indices possibles) sont choisies dans l'ordre croissant. L'indice des points a donc une grande importance. Par conséquent, les points sont réordonnés de façon à ce que ceux qui sont plus probables d'être représentants aient un indice plus faible. Nous utilisons l'heuristique FPF pour réordonner les points. On applique l'algorithme FPF en prenant $k = n$ (autant de classes que de points). Chaque point est donc choisi par l'algorithme FPF selon l'heuristique décrite en section 2, l'ordre de choix dans FPF donne l'ordre des points. Une autre heuristique utilisée est présentée dans [5].

Amélioration du modèle pour le critère du diamètre maximal Dans le cas sans contraintes définies par l'utilisateur, avec un k fixé, il est prouvé dans [13] que le diamètre d_{FPF} calculé par l'algorithme FPF vérifie $d_{opt} \leq d_{FPF} \leq 2d_{opt}$, avec d_{opt} le diamètre optimal. Cette connaissance implique des bornes pour la variable D, à savoir $[d_{FPF}/2, d_{FPF}]$. De plus, pour chaque couple de points i,j :
– si $d(i,j) < d_{FPF}/2$, la contrainte réifiée (1) sur i,j ne sera pas posée,
– si $d(i,j) > d_{FPF}$, la contrainte (1) est remplacée par une contrainte cannot-link $G[j] \neq G[i]$.

Ce résultat permet de supprimer plusieurs contraintes réifiées sans modifier la sémantique du modèle. Puisque les contraintes réifiées nécessitent la gestion de variables supplémentaires, leur suppression permet d'améliorer l'efficacité du modèle.

Dans le cas où s'ajoutent des contraintes d'utilisateur, le diamètre optimal est en général supérieur à d_{opt}. La borne supérieure du domaine de D n'est plus valable, par contre nous avons toujours la borne inférieure $d_{FPF}/2$. Ainsi, pour chaque couple i,j, si $d(i,j){<}d_{FPF}/2$, la contrainte (1) ne sera pas posée.

Algorithme de filtrage pour la somme des dissimilarités La contrainte (3) peut se réaliser par une contrainte linéaire sur des variables booléennes $V = \sum_{i<j} B_{ij} d(i,j)^2$, avec $B_{ij} = (G[i] == G[j])$. Cependant ces contraintes, considérées indépendamment, n'offrent pas suffisamment de propagation. Par exemple, prenons 4 points de 1 à 4, avec $k = 2$ et une affectation partielle où $G[1]{=}1$ et $G[2]{=}G[3]{=}2$. Nous avons trois variables booléennes B_{14}, B_{24}, B_{34} où $B_{i4} = (G[i] == G[4])$, avec $i \in [1,3]$. Supposons que pendant la recherche par *branch-and-bound*, la borne supérieure de V est mise à 4, donc avec la minimisation de V la contrainte $B_{14} + 2B_{24} + 3B_{34} < 4$ s'est

ajoutée. On constate que $B_{24} = B_{34}$ car $G[2] = G[3]$ et ne peuvent donc pas valoir 1, sinon la contrainte serait violée. On devrait conclure que le point 4 ne peut pas être dans le même cluster que les points 2 et 3, et supprimer ainsi la valeur 2 du domaine de $G[4]$. Ce filtrage n'est malheureusement pas fait pendant la propagation de la contrainte de somme. C'est le cas où des variables qui interviennent dans la contrainte de somme apparaissent également dans d'autres contraintes.

En PPC, des filtrages plus efficaces de la contrainte de somme ont été proposés, lorsqu'elle est considérée avec d'autres contraintes. Dans le cas d'une contrainte somme $y = \sum x_i$ avec des contraintes d'inégalité $x_j - x_i \leq c$, un algorithme peut réduire le domaine des x_i lorsque les bornes de y sont réduites [20]. Des algorithmes de réduction de bornes du domaine de variables sont proposés pour le cas d'une contrainte somme avec des contraintes d'ordre $x_i \leq x_{i+1}$ [18] ou avec une contrainte *alldifferent*$(x_1, \ldots, x_n)$ [2]. Cependant la contrainte (3) n'en fait pas partie. Un algorithme générique de réduction de borne qui exploite une contrainte somme et un ensemble de contraintes est proposé dans [19]. Dans notre cas, le domaine restant d'une variable $G[i]$ est constitué de valeurs qui sont des indices de représentants et n'est donc pas en général un intervalle continu. Nous avons donc implanté un algorithme de filtrage qui tient compte des connaissances de l'affectation partielle.

Supposons que l'ensemble des points soit séparé en deux ensembles disjoints K (les points i dont $G[i]$ est instanciée) et U (les points i dont $G[i]$ est non instanciée). Supposons que la contrainte (3) par le *branch-and-bound* devienne

$$\sum_{i \in K, j \in U} (G[i] == G[j]) a_{ij} + \sum_{i,j \in U} (G[i] == G[j]) a_{ij} < v$$

avec a_{ij} des constantes. L'idée est que pour chaque point inconnu $j \in U$, si l'affectation à un cluster c avec des points connus crée une valeur supérieure ou égale à v, alors la valeur c sera supprimée du domaine de $G[j]$. L'algorithme est comme suit :

pour chaque $j \in U$
  pour chaque *valeur c du domaine de* $G[j]$
    $m \leftarrow \sum_{i \in K, G[i]=c} a_{ij}$
    si $m \geq v$ **alors**
      supprimer c du domaine de $G[j]$

Cet algorithme n'est bien entendu pas complet car il ne tient compte que de la partie de la somme qui relie les points inconnus aux points connus. Il permet cependant d'améliorer l'efficacité du modèle pour pouvoir traiter la somme des dissimilarités avec un plus grand nombre de points.

4 Expérimentations

Des algorithmes classiques de classification non supervisée sont développés pour un critère particulier ou pour des types particuliers de contraintes définis par l'utilisateur. A cause de la complexité de la classification non supervisée, la plupart des algorithmes classiques se contentent d'un optimum local. De plus, la performance de ces méthodes diminue souvent lors de l'ajout de contraintes définies par l'utilisateur. Notre modèle est extensible à différents types de contraintes d'utilisateurs tout en traitant différents critères et garantit de trouver un optimum global.

Nous avons implanté notre modèle en utilisant la bibliothèque de programmation par contraintes Gecode version 3.7.3 [1]. Les expérimentations sont réalisées sur un processeur 2.4GHz Core i5 Intel sous Ubuntu 12.04. Toutes les contraintes sont réalisées directement par des contraintes de la bibliothèque Gecode, comme des contraintes `element`, `count` ou des contraintes réifiées. Nous présentons dans ce qui suit des résultats d'expérimentations sur des bases classiques et l'intérêt de la possibilité de combiner différents types de contraintes.

4.1 Clustering sans contraintes d'utilisateur

Nous avons expérimenté notre modèle en considérant différents schémas de classification, faisant varier le critère d'optimisation et considérant ou non des contraintes d'utilisateur. Nous considérons 5 bases de données du répertoire de données de UCI [2] :

Base de données	#Objets	#Classes
iris	150	3
ionosphere	351	2
synthetic control	600	6
vehicle	846	4
yeast	1484	10

Nous avons expérimenté avec k le nombre de classes connues. Le temps en secondes sur ces bases avec des critères de minimisation du diamètre maximal et de maximisation de la séparation minimale sont donnés dans le tableau suivant :

Base de données	Diamètre	Séparation
iris	0.1s	0.3s
ionosphere	0.8s	7.4s
synthetic control	24.6s	102.8s
vehicle	36.7s	308.6s
yeast	4211.2s	$>$ 2 heures

Le critère de diamètre est plus efficace dans tous les cas. Cela s'explique par notre stratégie de réordonner

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

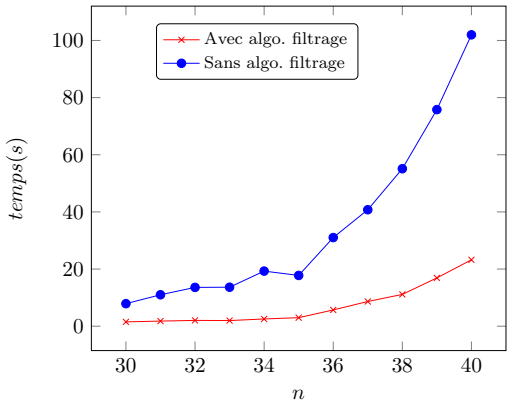

FIGURE 1 – Critère de somme des dissimilarités

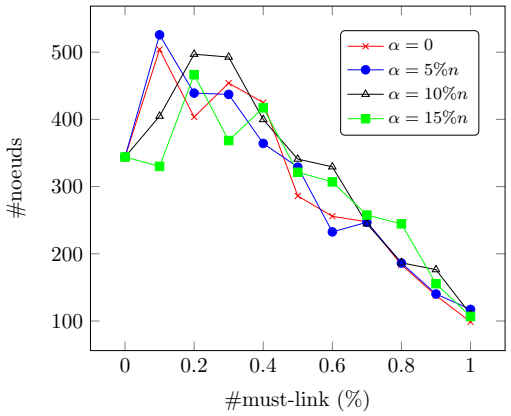

FIGURE 2 – Must-link et contrainte de capacité

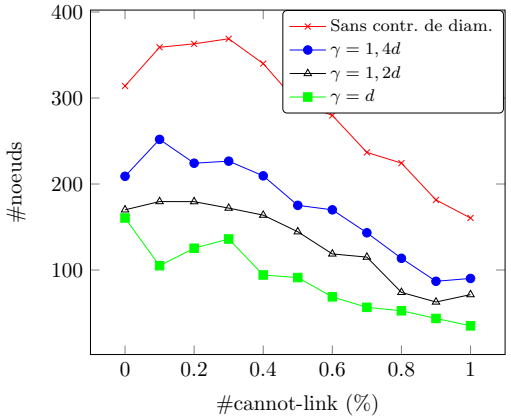

FIGURE 3 – Cannot-link et contrainte de diamètre

des points, appropriée pour ce critère : les k représentants de la solution optimale minimisant le diamètre maximal sont souvent les k premiers points après réordonnancement. Cette remarque n'est plus valable avec le critère de séparation. De plus, nous avons exploité des résultats théoriques pour améliorer le critère de diamètre.

Concernant le critère de minimisation de la somme des dissimilarités intraclasse, la propagation de contraintes est moins efficace. Il faut en général instancier suffisamment de points pour que la contrainte de somme puisse filtrer le domaine des variables restantes. Des expérimentations sont réalisées sur la base `iris`, avec un échantillon composé de 30 à 40 points. Figure 1 présente le temps en secondes nécessaire avec ou sans utilisation de l'algorithme de filtrage que nous avons implanté.

4.2 Clustering avec contraintes d'utilisateur

Concernant la classification non supervisée avec des contraintes d'utilisateur, nous présentons des expérimentations sur la base `iris`. Figure 2 présente le résultat de test où sont pris en compte le critère de minimisation du diamètre maximal, des contraintes must-link et une contrainte de capacité minimale. Le nombre de contraintes must-link varie de 0% à 1% du nombre de paires de points. La capacité minimale α testée est de 0 à 15% du nombre de points. Pour comparer les performances, nous choisissons de présenter le nombre de nœuds de l'arbre de recherche, car le temps est proche de 0 seconde. Nous constatons que des contraintes must-link améliore en général la performance, ce qui n'est pas le cas pour la contrainte de capacité.

Figure 3 présente le résultat de tests où sont pris en compte la maximisation de la séparation minimale, des contraintes cannot-link et une contrainte de diamètre. Le nombre de contraintes must-link varie de 0% à 1% du nombre de paires de points. Le paramètre γ de la contrainte de diamètre est de 1 à 1,4 fois le diamètre maximal d des classes réelles de la base. On constate que la contrainte cannot-link et la contrainte de diamètre augmente la performance.

Dans les tests de la Figure 4,on tient compte du critère de minimisation du diamètre maximal, d'une contrainte de séparation et d'une contrainte de densité. Le paramètre δ de la séparation varie de 0 à 10% de la distance maximale d entre deux points. Pour la contrainte de densité, $MinPts = 4$ et ϵ varie de 20% à 60% de d. On constate que la contrainte de séparation améliore la performance plus que la contrainte de densité. En effet, les états de recherche engendrés ne violent pas de contrainte de densité. En conséquence, la propagation de contrainte de densité réduit peu de nœuds dans l'arbre de recherche.

Nous avons également expérimenté le critère de minimisation de la somme des dissimilarités avec des contraintes must-link et cannot-link, dans les mêmes conditions que les tests de la Figure 1 avec notre

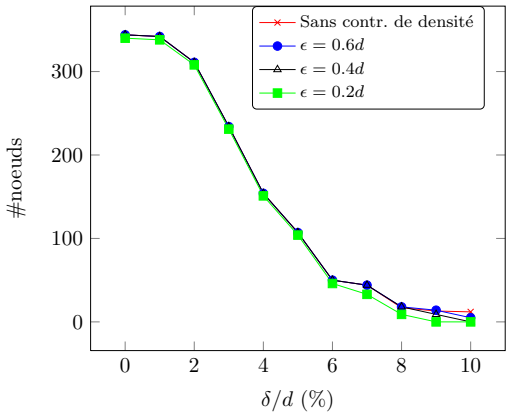

FIGURE 4 – Contraintes de séparation et de densité

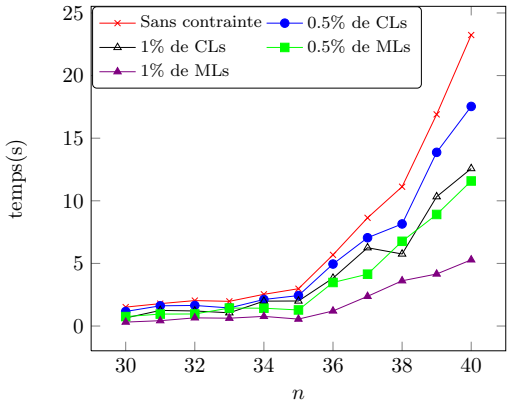

FIGURE 5 – Somme de dissimilarités avec ML et CL

contrainte globale. Le nombre de contraintes must-link ou cannot-link ajoutées varie de 0.5% à 1% du nombre de paires de points. Les résultats sont donnés en Figure 5. On constate que l'ajout des contraintes must-link améliore plus la performance car il a un impact plus direct sur la contrainte globale sur la somme.

4.3 Qualité de la solution

La possibilité de combiner différents types de contraintes permet de détailler plus finement la solution recherchée et en général permet d'améliorer la qualité de la solution trouvée. Nous considérons trois bases de données 2D similaires à celles utilisées dans [11]. Les données sont représentées dans la figure 6.

Dans la première base, on constate quatre groupes de diamètres différents. La deuxième base est plus difficile parce que les groupes ont des formes différentes. Avec le critère de diamètre maximal, le solveur trouve des groupes de diamètres plutôt homogènes, donc ce critère seul ne convient pas très bien, ainsi que l'illustre la Figure 7. L'ajout d'une contrainte de séparation,

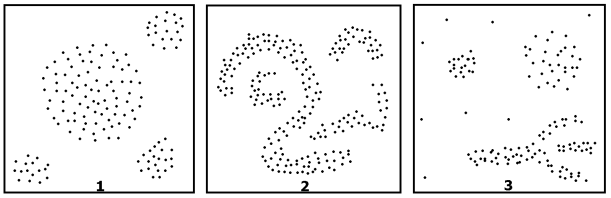

FIGURE 6 – Bases de données

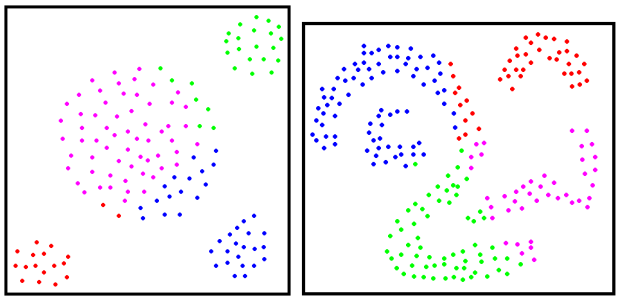

FIGURE 7 – Critère de diamètre

avec le paramètre δ valant 5% de la distance maximale entre paires de points, améliore nettement la qualité de la solution, comme le présente Figure 8. Notons que le critère de maximisation de la séparation minimale permet aussi de trouver cette solution.

La troisième base contient des points séparés (outliers). Notre modèle ne détecte pas encore les outliers, donc ces points sont aussi classés. Le critère de diamètre (Figure 9) ou de séparation (Figure 10) ne permet pas de trouver une solution de bonne qualité. Cependant, la qualité de la solution est nettement améliorée lorsqu'une contrainte de densité est ajoutée, avec $MintPts = 4$ et ϵ valant 25% de la distance maximale entre paires de points (Figure 11).

5 Conclusion

Nous présentons dans ce papier un modèle en PPC pour la classification non supervisée et celle

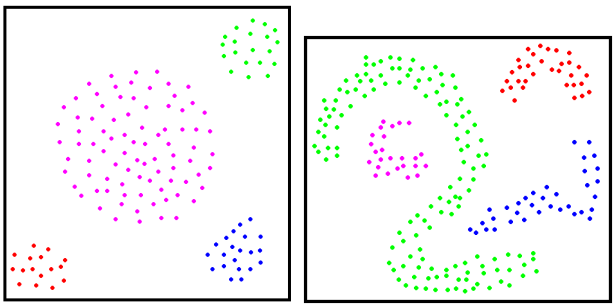

FIGURE 8 – Meilleure solution

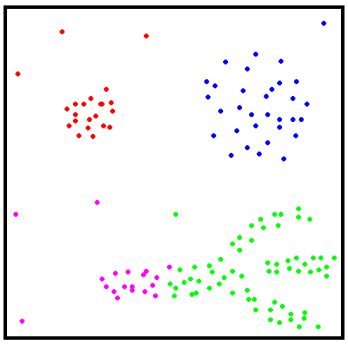

FIGURE 9 – Minimiser le diamètre maximal

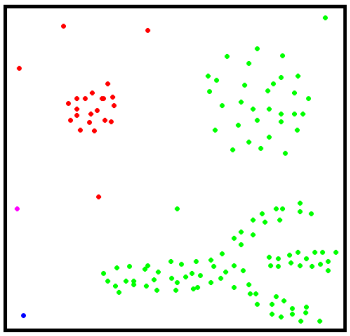

FIGURE 10 – Maximiser la séparation minimale

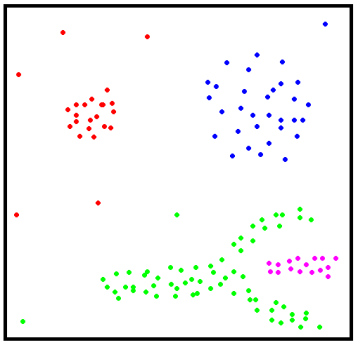

FIGURE 11 – Critère de séparation et contrainte de densité

sous contraintes d'utilisateur. Le modèle est générique dans le sens où il s'adapte à différents critères de classification et les contraintes d'utilisateurs les plus connues sont intégrées directement. Parmi des contraintes d'utilisateurs intégrées, nous généralisons une nouvelle contrainte portant sur la densité des clusters. Le modèle trouve un optimum global qui optimise un critère comme minimiser le diamètre maximal des clusters, maximiser le séparateur minimal entre clusters ou minimiser la somme des dissimilarités intra-classe. Du fait que le modèle se base sur la distance, des données qualitatives ou quantitatives peuvent être considérées. Des expérimentations sur des jeux de données classiques montrent l'intérêt de notre approche.

Le modèle actuel est plus efficace en traitant des critères qui peuvent se traduire par une relation entre deux points, comme le diamètre maximal ou la séparation minimale. Lorsque le critère se représente par une somme reliant un ensemble de points, comme la somme des dissimilarités intraclasse, la performance reste encore à améliorer. L'amélioration de l'efficacité constitue un objectif des travaux futurs afin de passer à l'échelle. Un des aspects envisagés est d'étudier une propagation de contraintes plus efficace pour exploiter des contraintes utilisateur.

Enfin, nous souhaitons également étendre le cadre pour traiter d'autres tâches de clustering, comme par exemple lorsque le nombre de classes n'est pas fixé à l'avance, ou la classification recouvrante.

Références

[1] Mihael Ankerst, Markus M. Breunig, Hans peter Kriegel, and Jörg Sander. Optics : Ordering points to identify the clustering structure. In *ACM SIGMOD international conference on Management of data*, pages 49–60. ACM Press, 1999.

[2] Nicolas Beldiceanu, Mats Carlsson, Thierry Petit, and Jean-Charles Régin. An o(nlog n) bound consistency algorithm for the conjunction of an alldifferent and an inequality between a sum of variables and a constant, and its generalization. In *ECAI 2012 - 20th European Conference on Artificial Intelligence*, pages 145–150, 2012.

[3] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st ICML*, pages 11–18, 2004.

[4] Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, Samir Loudni, and Jean-Philippe Métivier. Discovering Knowledge using a Constraint-based Language. *CoRR*, abs/1107.3407, 2011.

[5] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Une approche en PPC pour la classification non supervisée. In *13e EGC*, 2013.

[6] I. Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints : Theoretical and empirical results. *Proceedings of the 9th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 59–70, 2005.

[7] Ian Davidson and S. S. Ravi. Clustering with Constraints : Feasibility Issues and the k-Means Algorithm. In *Proc. 5th SIAM Data Mining Conference*, 2005.

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

[9] L. De Raedt, T. Guns, and S. Nijssen. Constraint Programming for Data Mining and Machine Learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[10] 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.

[11] Martin Ester, Hans P. Kriegel, Jorg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996. AAAI Press.

[12] F. Geraci, M. Pellegrini, M. Maggini, and F. Sebastiani. Cluster Generation and Cluster Labelling for Web Snippets : a Fast and Accurate Hierarchical Solution. In *Proceedings of the 13th Int. Conf. on String Processing and Information Retrieval*, pages 25–36, 2006.

[13] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38 :293–306, 1985.

[14] Tias Guns, Siegfried Nijssen, and Luc De Raedt. k-Pattern set mining under constraints. *IEEE Transactions on Knowledge and Data Engineering*, 2011.

[15] Zhengdong Lu and Miguel A. Carreira-Perpinan. Constrained spectral clustering through affinity propagation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2008.

[16] Jean-Philippe Métivier, Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, and Samir Loudni. Constrained Clustering Using SAT. In *IDA 2012, LNCS 7619*, pages 207–218, 2012.

[17] Marianne Mueller and Stefan Kramer. Integer linear programming models for constrained clustering. In *Discovery Science DS 2010*, pages 159–173, 2010.

[18] Thierry Petit, Jean-Charles Régin, and Nicolas Beldiceanu. A (n) bound-consistency algorithm for the increasing sum constraint. In *Principles and Practice of Constraint Programming CP 2011*, pages 721–728, 2011.

[19] Jean-Charles Régin and Thierry Petit. The objective sum constraint. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CPAIOR 2011*, pages 190–195, 2011.

[20] Jean-Charles Régin and Michel Rueher. Inequality-sum : a global constraint capturing the objective function. *RAIRO - Operations Research*, 39(2) :123–139, 2005.

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

[22] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. of the 18th ICML*, pages 577–584, 2001.

[23] Xiang Wang and Ian Davidson. Flexible constrained spectral clustering. In *KDD '10 : 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 563–572, 2010.