Comparaison de différents modèles de programmation par contraintes pour le clustering conceptuel

Maxime Chabert, Pierre-Antoine Champin, Amélie Cordier, Christine Solnon

Comparaison de différents modèles de programmation par contraintes pour le clustering conceptuel

Maxime Chabert[1,3]*Pierre-Antoine Champin[2] Amélie Cordier[2] Christine Solnon[1]

[1] LIRIS, INSA Lyon, Lyon, France
[2] LIRIS, Université Lyon 1, Lyon, France
[3] Infologic, Bourg-lès-Valence, France
{prénom.nom}@liris.cnrs.fr

Résumé

Le clustering conceptuel permet de partitionner un ensemble d'objets en clusters d'objets similaires, correspondant à des concepts formels. Nous présentons une nouvelle approche basée sur la programmation par contraintes, où l'ensemble des concepts formels est extrait dans une étape de pré-traitement en utilisant des techniques spécialisées de fouille de données. Nous comparons l'efficacité de notre approche avec celle de plusieurs approches récentes utilisant la programmation par contraintes ou la programmation linéaire en nombres entiers sur des instances classiques d'apprentissage automatique. Nous introduisons également un nouvel ensemble d'instances provenant d'une application réelle, visant à extraire des concepts de paramétrage à partir de séquences de paramétrage d'un progiciel de gestion, et nous évaluons la pertinence des concepts extraits en fonction des critères utilisés dans la définition de la fonction objectif.

Abstract

Conceptual clustering allows to partition a set of objects into clusters of similar objects, corresponding to formal concepts. We present a new approach based on constraint programming where formal concepts are extracted in a pre-processing step by using a dedicated data-mining approach. We compare the efficiency of our approach with several recent approaches using constraint programming or integer linear programming on classical machine learning instances. We also introduce a new set of instances coming from a real application case, which aims at extracting setting concepts from an Enterprise Ressource Planning (ERP) software. We assess the relevance of extracted concepts depending on criteria used in the objective function.

*Papier doctorant : Maxime Chabert[1,3] est auteur principal.

1 Introduction

Nous proposons dans cet article de nouveaux modèles à base de contraintes pour résoudre un problème de clustering conceptuel, et nous évaluons ces modèles sur des instances académiques, ainsi que sur un nouvel ensemble d'instances provenant d'une application réelle visant à automatiser la phase de paramétrage d'un progiciel de gestion (*Enterprise Ressource Planning*, ERP).

Présentation du contexte applicatif du travail. Les ERP sont des logiciels avec un vaste périmètre fonctionnel allant de la gestion commerciale jusqu'à la gestion des ateliers de production et de stockage [8]. Cette amplitude de fonctionnalités rend le processus d'installation complexe, et une étude récente souligne que 57% des installations d'ERP dépassent le budget et le temps prévus [1]. Nous avons étudié le processus d'installation du progiciel de gestion Copilote de la société Infologic, spécialisée en agro-alimentaire. Il apparaît que 65% du temps est dédié à la phase de paramétrage : cette phase consiste à affecter des valeurs à des paramètres afin de répondre aux besoins du client et à ses spécificités structurelles et organisationnelles [18]. Cette complexité est due au grand nombre de paramètres pouvant interagir entre eux. De plus, plusieurs études [14, 2] portant sur le processus d'installation des ERP montrent que le paramétrage n'est pas considéré comme un facteur critique de succès contrairement à l'accompagnement au changement ou à la formation des utilisateurs. C'est pourquoi, réduire le temps de paramétrage devient un réel enjeu pour les

intégrateurs d'ERP afin d'allouer plus de temps aux phases critiques du processus d'installation.

Pour répondre à cette problématique, nous proposons d'analyser une base de paramétrages existants, correspondant à des installations de l'ERP chez différents clients. Notre objectif est d'identifier des séquences pertinentes de paramétrage afin de les associer à des besoins fonctionnels. Étant donné que beaucoup de besoins se retrouvent chez plusieurs clients, ces séquences de paramétrage pourront être réutilisées durant l'installation de l'ERP chez un nouveau client ayant des besoins similaires.

Pour identifier des séquences pertinentes de paramétrage, nous proposons de partitionner la base de paramétrages de façon à regrouper les paramétrages similaires. Nous proposons pour cela d'utiliser le clustering conceptuel [12] car cette approche ne pré-suppose pas qu'il existe une fonction de similarité permettant d'évaluer la similarité de deux objets : chaque cluster correspond à un concept formel et est décrit par l'ensemble des paramètres communs à tous les paramétrages du cluster.

Présentation des contributions de l'article. Plusieurs travaux récents proposent de résoudre des problèmes de clustering conceptuel en utilisant des approches déclaratives telles que la programmation par contraintes (PPC) [4] ou la programmation linéaire en nombres entiers (PLNE) [15]. Ces approches sont particulièrement pertinentes dans notre contexte applicatif du fait de leur souplesse pour ajouter des contraintes ou modifier la fonction objectif : une de nos problématiques majeures est de trouver une fonction objectif et des contraintes permettant d'extraire des concepts de paramétrage pertinents pour les experts métiers.

Nous présentons une nouvelle approche basée sur la PPC. Comme proposé dans [15], nous introduisons une étape de pré-traitement pour extraire l'ensemble des concepts formels candidats à l'aide d'un outil dédié au problème de l'extraction de motifs fréquents. Nous proposons d'utiliser la programmation par contraintes pour sélectionner un sous-ensemble de cet ensemble formant une partition optimale, et nous introduisons deux nouveaux modèles ensemblistes pour cela. Nous comparons ces modèles avec ceux de [15] et [4]. Cette comparaison est réalisée sur un ensemble d'instances classiques dans le domaine de l'apprentissage automatique. Nous introduisons également un nouveau benchmark composé d'instances construites à partir de notre base de paramétrages. Enfin, nous comparons d'un point de vue qualitatif la qualité des clusterings obtenus selon différentes fonctions objectifs.

Organisation de l'article. La section 2 définit formellement le problème de clustering conceptuel, et décrit les approches permettant de résoudre ce problème, et plus particulièrement les approches récentes de [15] et [4]. La section 3 introduit deux nouveaux modèles PPC pour résoudre ce problème, l'un basé sur les clusters, l'autre sur les transactions. Enfin, la section 4 compare les différentes approches en termes de passage à l'échelle, et la section 5 compare la qualité des solutions calculées en fonction des critères considérés dans la fonction objectif.

2 Contexte

2.1 Clustering conceptuel

Le clustering conceptuel est une approche de classification non-supervisée qui vise à partitionner un ensemble d'objets en clusters homogènes. La particularité de cette méthode est qu'elle donne, en plus des clusters, une description de chaque cluster sous la forme d'un concept formel.

Soit $\mathcal{T}$ un ensemble de m transactions (ou objets), $\mathcal{I}$ un ensemble de n items (ou attributs), et $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{I}$ une relation binaire qui lie les transactions aux items : $(t, i) \in \mathcal{R}$ (noté également $t\mathcal{R}i$) traduit le fait qu'une transaction t contient l'item i. Nous supposons que toutes les transactions ont des ensembles d'items différents, *i.e.*,

$$\forall t, t' \in \mathcal{T}, t \neq t' \Rightarrow \{i \in \mathcal{I} : t\mathcal{R}i\} \neq \{i \in \mathcal{I} : t'\mathcal{R}i\}.$$

Étant donné un ensemble E, nous notons $\mathcal{P}(E)$ l'ensemble de ses sous-ensembles, et $\#E$ sa cardinalité. Enfin, sans perte de généralité, nous supposons que les transactions sont numérotées de 1 à m et les items de 1 à n.

L'*intention* d'un sous-ensemble $T \subseteq \mathcal{T}$ de transactions est l'ensemble des items contenus dans toutes les transactions de T, *i.e.*,

$$intent(T) = \{i \in \mathcal{I} : \forall t \in T, t\mathcal{R}i\}.$$

L'*extension* d'un sous-ensemble $I \subseteq \mathcal{I}$ d'items est l'ensemble des transactions qui contiennent tous les items de I, *i.e.*,

$$extent(I) = \{t \in \mathcal{T} : \forall i \in I, t\mathcal{R}i\}.$$

Ces deux opérateurs induisent une connexion de Galois entre $\mathcal{P}(\mathcal{T})$ et $\mathcal{P}(\mathcal{I})$, *i.e.*,

$$T \subseteq extent(I) \Leftrightarrow I \subseteq intent(T).$$

Un *concept formel* est un couple (T, I) avec $T \subseteq \mathcal{T}$ et $I \subseteq \mathcal{I}$ tels que $T = extent(I)$ et $I = intent(T)$. On

TABLE 1 – Jeu de données transactionnel $\mathcal{T}$

	i_1	i_2	i_3	i_4
t_1	1	0	0	1
t_2	1	0	1	1
t_3	0	1	0	1
t_4	0	1	1	0
t_5	1	0	1	0

note $\mathcal{F}$ l'ensemble de tous les concepts formels. Notons qu'un concept formel correspond à un ensemble clos d'items (*closed itemset*) tel que défini en fouille de données. Par conséquent, l'ensemble $\mathcal{F}$ des concepts formels peut être calculé en utilisant un algorithme de recherche de motifs clos fréquents (tel que LCM [19], par exemple), en fixant le seuil de fréquence à 1.

En clustering conceptuel, chaque cluster correspond à un concept formel, et un clustering est un ensemble de k concepts formels $\mathcal{C} = \{(T_1, I_1), \ldots, (T_k, I_k)\}$ tel que $\{T_1, \ldots, T_k\}$ forme une partition de l'ensemble de transactions $\mathcal{T}$.

La fréquence d'un cluster (T_j, I_j) est son nombre de transactions, *i.e.*, $freq(T_j, I_j) = \#T_j$, et sa taille est son nombre d'items, *i.e.*, $taille(T_j, I_j) = \#I_j$.

Différents critères peuvent être considérés pour définir la qualité d'un clustering conceptuel. Dans cet article, nous en considérons trois :

1. maximiser la taille minimale d'un cluster, de façon à éviter d'avoir des concepts comportant peu d'items ;
2. maximiser la fréquence minimale d'un cluster, de façon à éviter d'avoir des concepts comportant peu de transactions ;
3. maximiser la somme des tailles des clusters, de façon à favoriser les concepts comportant un grand nombre d'items.

Exemple. La table 1 présente un jeu de données transactionnelles $\mathcal{T}$ composé de cinq transactions définies sur quatre items. La table 2 donne l'ensemble $\mathcal{F}$ des concepts formels de $\mathcal{T}$. Par exemple, le concept c_1 est défini par le couple $(\{i_1\}, \{t_1, t_2, t_5\})$. Sa fréquence et sa taille sont : $freq(c_1) = 3$ et $taille(c_1) = 1$. $\mathcal{C}_1 = \{(\{i_1\}, \{t_1, t_2, t_5\}), (\{i_2\}, \{t_3, t_4\})\}$ et $\mathcal{C}_2 = \{(\{i_1\}, \{t_1, t_2, t_5\}), (\{i_2, i_4\}, \{t_3\}), (\{i_2, i_3\}, \{t_4\})\}$ sont deux exemples de clusterings de $\mathcal{T}$. Selon le critère considéré, la qualité de $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) est évaluée à 1 (resp. 1), pour le critère de taille minimale d'un cluster, 2 (resp. 1), pour le critère de fréquence minimale d'un cluster, et 2 (resp. 5), pour le critère de somme des tailles des clusters.

TABLE 2 – Ensemble $\mathcal{F}$ des concepts formels de $\mathcal{T}$

C	*intent*	*extent*	*fréq.*	*taille*
c_1	$\{i_1\}$	$\{t_1, t_2, t_5\}$	3	1
c_2	$\{i_3\}$	$\{t_2, t_4, t_5\}$	3	1
c_3	$\{i_1, i_3\}$	$\{t_2, t_5\}$	2	2
c_4	$\{i_4\}$	$\{t_1, t_2, t_3\}$	3	1
c_5	$\{i_1, i_4\}$	$\{t_1, t_2\}$	2	2
c_6	$\{i_1, i_3, i_4\}$	$\{t_2\}$	1	3
c_7	$\{i_2\}$	$\{t_3, t_4\}$	2	1
c_8	$\{i_2, i_3\}$	$\{t_4\}$	1	2
c_9	$\{i_2, i_4\}$	$\{t_3\}$	1	2

2.2 Approches dédiées au clustering conceptuel

Depuis l'introduction du clustering conceptuel par [12], de multiples approches dédiées à ce problème ont été proposées.

Plusieurs de ces approches utilisent des heuristiques basées sur des mesures statistiques pour construire des clusters [7] pouvant être organisés en hiérarchie [5, 10]. Le système COBWEB [5] s'appuie sur la similarité intra-cluster et la dissimilarité inter-cluster pour construire incrémentalement une hiérarchie de clusters, l'interprétation conceptuelle des clusters étant alors une étape indépendante de la construction des clusters. Ces deux tâches sont souvent découplées dans des approches plus récentes soit en utilisant des techniques de clustering après avoir extrait un ensemble de concepts [16], soit en clusterisant des objets puis en extrayant une description associée à chaque cluster [17]. D'autres approches ont introduit l'utilisation de connaissances lors de la construction des clusters pour améliorer la pertinence des concepts extraits [13, 9, 20].

La qualité des résultats obtenus par ces approches reste variable tout comme le passage à l'échelle sur de grands volumes de données.

Ces approches dédiées ne permettent pas facilement d'ajouter de nouvelles contraintes, ou de modifier la fonction objectif. Ce point étant particulièrement important dans notre contexte applicatif où nous souhaitons évaluer la qualité de différents clusterings obtenus en considérant différents critères, nous nous sommes intéressés à des approches déclaratives telles que la PPC et la PLNE.

2.3 PPC pour le clustering conceptuel

Guns a montré dans sa thèse [6] que la PPC fournit un cadre déclaratif permettant de facilement modéliser différents problèmes de recherche de motifs fréquents, et que les solveurs génériques de PPC peuvent être compétitifs avec des algorithmes dédiés. Guns a

notamment proposé une modélisation utilisant des variables binaires pour exprimer le fait qu'un item appartient à l'intention d'un concept associé à un cluster. Dao et al [4] ont proposé un nouveau modèle utilisant des variables ensemblistes, et ont montré que ce modèle ensembliste a de bien meilleures performances en pratique que le modèle binaire. Nous décrivons ici ce modèle ensembliste.

Variables. Pour chaque transaction $t \in \mathcal{T}$, la variable entière G_t représente le cluster de t. Le nombre de clusters est défini par une constante k donnée en entrée, et les clusters sont numérotés de 1 à k. Ainsi, chaque variable G_t a pour domaine $D(G_t) = [1, k]$.

Pour chaque cluster $c \in [1, k]$, la variable ensembliste E_c représente l'ensemble des items de l'intention du concept formel associé au cluster c. Le domaine de E_c est l'ensemble des sous-ensembles de $\mathcal{I}$, *i.e.*, $D(E_c) = \mathcal{P}(\mathcal{I})$.

Contraintes. Les symétries (dues au fait que les clusters sont interchangeables) sont éliminées en posant une contrainte de précédence [11] :

$$precede(G, [1, k]).$$

Cette contrainte assure que la première transaction appartient au premier cluster (*i.e.*, $G_1 = 1$), et que $\forall j \in [2, n], \exists l < j, G_l = G_j - 1$.

Chaque cluster est contraint à posséder au moins une transaction à l'aide de la contrainte :

$$atLeast(1, G, k).$$

La contrainte d'extension est exprimée par :

$$\forall c \in [1, k], \forall t \in \mathcal{T}, G_t = c \Leftrightarrow E_c \subseteq \{i \in \mathcal{I} | t \mathcal{R} i\}.$$

La contrainte d'intention est exprimée par :

$$\forall c \in [1, k], E_c = \cap_{t \in \mathcal{T}, G_t = c} \{i \in \mathcal{I} | t \mathcal{R} i\}.$$

Chaque contrainte d'intention nécessite n contraintes de domaine réifiées pour construire l'ensemble $I_c = \{t \in \mathcal{T} | G_t = c\}$, et une contrainte *element* ensembliste.

Fonction objectif. Pour maximiser la fréquence minimale des clusters, on introduit une variable entière F devant être maximisée. Son domaine est $D(F) = [1, m]$, et elle est contrainte à être inférieure ou égale à la fréquence de chaque cluster $c \in [1, k]$ en posant une contrainte $atLeast(F, G, c)$.

Pour maximiser la taille minimale des clusters, on introduit une variable entière T devant être maximisée. Son domaine est $D(T) = [1, n]$, et elle est contrainte à être inférieure ou égale à la taille de chaque cluster $c \in [1, k]$ en posant la contrainte $T \leq \#E_c$.

Si le cas n'a pas été explicitement étudié dans [4], on peut facilement étendre ce modèle pour maximiser la somme des tailles en ajoutant une variable S devant être maximisée. Son domaine est $D(S) = nk$, et elle est contrainte à être égale à la somme des variables T.

Extension du modèle à un nombre variable de clusters. Le modèle introduit dans [4] suppose que le nombre de clusters est fixé par une constante k. L'extension au cas où le nombre de clusters n'est pas connu *a priori* est relativement triviale : il suffit d'introduire une constante $kMax$, fixant le nombre maximal de clusters (si le nombre de clusters n'est pas borné, alors $kMax = m$), et de définir k comme une variable entière de domaine $D(k) = [2, kMax]$. Dans ce cas, la contrainte $atLeast(1, G, k)$ n'a plus de raison d'être.

2.4 PLNE pour le clustering conceptuel

Ouali et al [15] ont proposé de combiner un outil dédié à l'extraction de motifs fréquents avec la PLNE pour faire du clustering conceptuel : dans une étape de pré-traitement, l'ensemble de tous les concepts formels est calculé en utilisant un outil dédié à ce problème tel que LCM [19] ; la PLNE est utilisée ensuite pour sélectionner un sous-ensemble de ces concepts qui forme une partition de $\mathcal{T}$ et qui optimise la fonction objectif.

Plus précisément, soit $\mathcal{F}$ l'ensemble de tous les concepts formels calculés en pré-traitement. L'objectif est de sélectionner un sous-ensemble de $\mathcal{F}$ tel que chaque transaction de $\mathcal{T}$ appartienne à exactement un concept formel du sous-ensemble, et optimise un critère donné. Cela est modélisé en PLNE dans [15] de la façon suivante.

Variables. Pour chaque concept formel $f \in \mathcal{F}$, on introduit une variable binaire x_f telle que $x_f = 1$ ssi le concept formel f est sélectionné.

La variable entière k correspond au nombre de concepts sélectionnés.

Contraintes. Pour garantir que l'ensemble des concepts sélectionnés forme une partition de $\mathcal{T}$ on pose, pour chaque transaction $t \in \mathcal{T}$, la contrainte :

$$\sum_{f \in \mathcal{F}} a_{tf} x_f = 1.$$

où $a_{tf} = 1$ si la transaction t appartient à l'extension du concept f.

Pour contraindre k à être égal au nombre de concepts sélectionnés, on pose la contrainte :

$$k = \sum_{f \in \mathcal{F}} x_f.$$

Enfin, on peut borner le nombre de concepts sélectionnés k en posant la contrainte :

$$kMin \leq k \leq kMax.$$

Fonction objectif. Un gain v_f est associé à chaque concept formel $f \in \mathcal{F}$: v_f est égal à la taille de f. La fonction objectif à maximiser est la somme des gains :

$$\sum_{f \in \mathcal{F}} v_f x_f.$$

Si le cas n'a pas été explicitement étudié dans [15], on peut facilement étendre le modèle pour maximiser le gain minimal d'un concept. Le gain v_f est égal soit à la taille de f, soit à sa fréquence, selon le critère choisi. Une variable v_{min} est introduite et est contrainte à être inférieure ou égale au gain des concepts sélectionnés en posant, pour chaque concept formel $f \in \mathcal{F}$, la contrainte :

$$v_{min} \leq v_f x_f + M(1 - x_f)$$

où M est une constante positive supérieure au plus grand gain possible. La fonction objectif à maximiser est v_{min}.

3 Nouveaux modèles PPC

Nous proposons de nous inspirer de l'approche de [15], consistant à extraire dans une phase de prétraitement l'ensemble $\mathcal{F}$ de tous les concepts formels avec un outil dédié à ce problème, et nous proposons d'évaluer les capacités de la programmation par contraintes pour sélectionner le sous-ensemble de concepts formels formant un clustering optimal.

Nous proposons deux modèles utilisant des contraintes ensemblistes : le premier modèle associe une variable entière à chaque cluster (déterminant le concept formel associé au cluster) et pose une contrainte ensembliste de partition sur l'ensemble des extensions des concepts formels associés aux clusters ; le second modèle utilise une variable ensembliste pour représenter le sous-ensemble de clusters sélectionnés et pose des contraintes *member* et *card* pour assurer que ce sous-ensemble définit bien une partition.

3.1 Modèle ensembliste basé sur les clusters

Variables. Pour chaque cluster $c \in [1, kMax]$, on définit une variable entière G_c déterminant le concept formel associé au cluster c. Comme la solution optimale peut avoir moins de $kMax$ clusters, on introduit un concept formel vide : ce concept a le numéro 0, et son extension est l'ensemble vide, *i.e.*, $extent(0) = \emptyset$. Nous supposons que les concepts formels de $\mathcal{F}$ sont numérotés de 1 à p. Par conséquent, le domaine de chaque variable G_c est $D(G_c) = [0, p]$: si $G_c \in [1, p]$, alors le cluster c correspond au concept formel G_c ; si $G_c = 0$ alors le cluster c est vide.

On introduit une variable entière k qui représente le nombre de clusters non vides de la solution, et une autre variable entière k_{empty} qui représente le nombre de clusters vides de la solution. Leurs domaines sont, respectivement, $D(k) = [2, kMax]$ et $D(k_{empty}) = [0, kMax - 2]$.

Contraintes. Pour éliminer les symétries, dues au fait que les valeurs affectées à deux variables G_i et G_j peuvent être interchangées, nous contraignons G à prendre des valeurs croissantes par rapport à un ordre défini sur $\mathcal{F}$. Pour assurer que les extensions des clusters forment une partition de l'ensemble des transactions, nous posons une contrainte de partition [3] :

$$partition(\{extent(G_c) | c \in [1, kMax]\}, [1, m])$$

Pour assurer que le nombre de clusters vides est égal à k_{empty}, nous posons la contrainte :

$$count(G, 0, k_{empty})$$

Enfin, nous assurons que k est égal au nombre de clusters non vides à l'aide de la contrainte :

$$k + k_{empty} = kMax.$$

Fonction objectif. Un gain v_f est associé à chaque concept formel $f \in \mathcal{F}$. Selon les cas, ce gain peut être la fréquence ou la taille de f. La fonction objectif à maximiser peut être soit la somme des gains, *i.e.*,

$$\sum_{c=1}^{kMax} v_{G_c}$$

(et dans ce cas on définit le gain du cluster vide par $v_0 = 0$), soit le gain minimal, *i.e.*,

$$\min_{c \in [1, kMax]} v_{G_c}$$

(et dans ce cas on définit le gain du cluster vide par $v_0 = \infty$).

Stratégie de recherche. Deux stratégies de recherches différentes sont utilisées selon si la taille ou la fréquence est utilisée comme critère de la fonction

objectif. Dans les deux cas, les concepts formels sont classés par ordre décroissant selon leur gain v_f. Ainsi, $\forall f, f' \in F, f > f' \Leftrightarrow v_f < v_{f'}$.

Si le critère considéré est la taille, les solutions tendent à avoir un nombre de clusters proche de $kMax$. Nous utilisons un sélecteur qui choisit la borne minimale du domaine comme prochaine valeur pour la variable k_{empty}. De la même manière, pour chaque variable G_c, les concepts formels de plus grande taille sont d'abord choisis. Ainsi, les solutions avec un maximum de clusters non vides sont recherchées en premier.

Si le critère considéré est la fréquence, les solutions tendent à avoir peu de clusters. Les variables de décision sont uniquement les variables G_c. Nous utilisons la stratégie *first fail* qui consiste à sélectionner la variable avec le plus petit domaine comme prochaine variable à instancier. De plus, le sélecteur des variables G_c choisit la borne minimale du domaine comme prochaine valeur. Ainsi, les solutions avec le moins de clusters possibles sont explorées en premier.

3.2 Modèle ensembliste basé sur les transactions

Variables. Pour chaque transaction $t \in \mathcal{T}$, nous définissons une variable entière C_t déterminant le concept sélectionné dont l'extension contient t : chaque transaction t doit appartenir à l'extension d'exactement un concept sélectionné, et l'ensemble des concepts candidats est l'ensemble des concepts dont l'extension contient t. Ainsi, pour chaque transaction $t \in \mathcal{T}$, le domaine de C_t est $D(C_t) = \{f \in F \mid t \in extent(f)\}$.

Nous définissons une variable ensembliste P déterminant l'ensemble des concepts sélectionnés : chaque concept appartenant à P correspond à un cluster. Le domaine de P est : $D(P) = \mathcal{P}(\mathcal{F})$.

La variable entière k définit le nombre de clusters de la solution (et donc la cardinalité de P). Son domaine est : $D(k) = [2, kMax]$.

Contraintes. On assure que, pour chaque transaction $t \in \mathcal{T}$, C_t est un élément de l'ensemble P en posant la contrainte :

$$member(C_t, P)$$

On assure que, pour chaque transaction $t \in \mathcal{T}$, il y a exactement un concept formel de P dont l'extension contient t (autrement dit, les extensions des concepts de P forment une partition de l'ensemble des transactions) en vérifiant que l'intersection entre P et l'ensemble des concepts dont t appartient à l'extension contient un seul élément, *i.e.*,

$$\forall t \in \mathcal{T}, card(\{f \in F \mid t \in extent(f)\} \cap P) = 1$$

Enfin, le nombre de clusters de la solution est contraint avec la contrainte $card(P) = k$, assurant que le nombre d'éléments de P est égal à k, *i.e.*, le nombre de clusters est égal à k.

Fonction objectif. Comme pour le modèle précédent, la fonction objectif est définie en associant un gain v_f à chaque concept formel f.

Stratégie de recherche. Les concepts formels sont classés par ordre décroissant selon leur gain v_f. Les variables de décision sont uniquement les variables C_t avec une stratégie *first fail*. De plus, le sélecteur des variables C_t choisit la borne minimale du domaine comme prochaine valeur.

4 Comparaison expérimentale des différents modèles

Dans cette section, nous comparons l'efficacité de nos modèles par rapport aux approches de Ouali et al. [15] et de Bich et. al [4].

Protocole expérimental. Toutes les expérimentations ont été menées sur un Intel(R) Core(TM) i7-6700 avec 3.40GHz de CPU et 65GB de RAM. Nous avons utilisé LCM [19] pour extraire les concepts formels, Gecode v4.3 pour les modèles PPC et Cplex v12.7 pour le modèle PLNE. Dans chaque modèle, nous avons fixé $kMax$, le nombre de cluster maximal, à $m-1$, m étant le nombre de transactions de l'instance. Chaque résolution a été limitée à deux heures de temps CPU.

Description des instances. Nous avons considéré quatre instances classiques en apprentissage automatique et utilisées dans [15] : zoo, vote, tic-tac-toe et mushroom. Nous avons également considéré quatre

TABLE 3 – Description des jeux de données : chaque ligne donne successivement le nom du jeu de données, le nombre de transactions, le nombre d'items, le nombre de concepts, et le temps (en secondes) mis par LCM pour extraire les concepts.

Nom	$\#\,\mathcal{T}$	$\#\,\mathcal{I}$	$\#\,\mathcal{F}$	Temps
ERP 1	50	27	1 580	0,01
ERP 2	84	42	14 305	0,05
ERP 3	95	61	71 918	0,45
ERP 4	160	66	728 537	5,31
zoo	59	36	4 567	0,01
vote	341	48	227 031	0,54
tic-tac-toe	958	27	42 711	0,05
mushroom	8 124	119	221 524	3,85

TABLE 4 – Comparaison des temps de résolution : chaque ligne donne successivement le nom de l'instance, et les résultats pour les trois critères à maximiser (taille minimale, fréquence minimale, et somme des tailles). Pour chaque critère, nous donnons les temps CPU (en secondes) des trois modèles PPC (FullCP, CB et TB) et du modèle PLNE (LP). Pour les modèles CB, TB et LP, le temps CPU comprend le temps mis par LCM pour extraire les concepts formels. Le symbole "-" indique que le temps dépasse la limite de 2h.

Instance	(1) - Max. taille minimale				(2) - Max. fréquence minimale				(3) - Max. somme des tailles			
	FullCP	CB	TB	LP	FullCP	CB	TB	LP	FullCP	CB	TB	LP
ERP 1	0,6	0,3	**0,0**	0,4	**0,1**	0,2	**0,1**	0,4	-	-	-	**0,1**
ERP 2	6,3	4,8	**0,4**	3,7	3,5	6,1	**0,6**	14,4	-	-	-	**1,1**
ERP 3	5,9	24,0	**2,6**	2 356,9	12,0	282,6	**6,7**	159,9	-	-	-	**7,3**
ERP 4	74,6	457,9	**35,6**	-	1 613,2	-	**204,6**	-	-	-	-	**92,0**
zoo	1,7	0,8	**0,1**	0,2	0,4	0,4	**0,1**	2,5	-	-	-	**0,2**
vote	2885,6	1478,0	**46,6**	-	-	-	**10,2**	-	-	-	-	**29,2**
tic-tac-toe	-	-	**240,4**	484,6	1 115,1	5 982,0	**10,1**	684,9	-	-	-	**5,7**
mushroom	-	-	-	-	-	-	**1 577,5**	-	-	-	-	-

instances construites à partir de notre base de paramétrages. Cette base comporte 400 paramétrages, chaque paramétrage correspondant à une installation de l'ERP Copilote chez un client différent. Chacun de ces paramétrages spécifie les valeurs de près de 450 paramètres (chaque paramètre pouvant prendre un nombre fini de valeurs différentes). Nous avons transformé chaque couple paramètre/valeur en un item booléen et extrait quatre instances de tailles différentes pour pouvoir évaluer plus finement le passage à l'échelle des différentes approches considérées. La table 3 présente les caractéristiques de chaque instance ainsi que le temps d'extraction des concepts formels avec LCM. Nous pouvons remarquer que ce temps dépend du nombre de concepts formels (la complexité de LCM est linéaire par rapport à $\#\mathcal{F}$), et est relativement court. Par exemple, les 728 537 concepts formels de l'instance ERP 4 sont extraits en moins de six secondes.

Modèles comparés. Nous avons comparé les deux modèles décrits dans la partie 2.2, à savoir l'approche de Ouali et al. [15] (appelée LP) et celle de Dao et al. [4] (appelée FullCP), avec nos deux modèles introduits dans la partie 3 : le modèle basé sur les clusters (appelé CB) et celui basé sur les transactions (appelé TB).

La table 4 compare les temps de résolution des différents modèles pour les trois critères d'optimisation considérés. Pour les modèles CB, TB et LP, le temps CPU comprend le temps de pré-traitement (*i.e.*, le temps mis par LCM pour extraire les concepts formels).

Temps de résolution pour les critères (1) et (2). Quand le critère considéré est la maximisation du minimum de la taille (1) ou de la fréquence (2), le modèle TB domine tous les autres modèles sur l'ensemble des instances. Il est souvent un ordre de grandeur plus rapide que les autres modèles. Cependant, il ne trouve pas la solution optimale dans le temps imparti pour l'instance mushroom pour le minimum de la taille (1). Cela s'explique probablement par le nombre de transactions plus important (10^3) comparé aux autres instances (de 10^1 à 10^2).

La différence de performance du modèle TB avec le modèle CB est probablement due à l'efficacité de la stratégie *first fail* qui permet de choisir la variable C_t ayant le moins de concepts formels dans son domaine. Dans les deux modèles, l'espace des concepts candidats (*i.e.*, les concepts dont l'extension n'a aucune transaction appartenant aux extensions des concepts déjà sélectionnés) est réduit de la même manière par la contrainte de partition. En revanche, le modèle TB choisit en priorité les concepts contenus dans la transaction ayant le moins de candidats (contrairement au modèle CB), et réduit ainsi efficacement l'espace de recherche pour converger plus rapidement vers la solution optimale. De plus, l'heuristique de CB sur le nombre de clusters vides n'est pas toujours efficace, comme par exemple pour tic-tac-toe avec le critère (2).

Nous pouvons noter les bonnes performances de FullCP sur les instances ERP : FullCP est capable de résoudre toutes ces instances (pour les critères (1) et (2)), et il est souvent plus rapide que CB et LP. En revanche, il n'est capable de résoudre que la moitié des quatre instances académiques.

Enfin, le modèle LP est toujours moins efficace que TB et ne parvient pas à résoudre trois instances (ERP 4, vote et mushroom) pour les critères (1) et (2). Le plus grand nombre de concepts formels de ces instances (10^5 contre 10^4 et 10^3) explique probablement cette contre-performance.

TABLE 5 – Temps CPU (en secondes) pour trouver la solution optimale avec le modèle TB quand le critère est la maximisation de la somme des tailles (3).

Instance	Temps
ERP 1	0,1
ERP 2	1,1
ERP 3	9,4
ERP 4	815,8
zoo	0,3
vote	831,2
tic-tac-toe	930,4

Temps de résolution pour le critère (3). Quand le critère considéré est la maximisation de la somme des tailles, la seule approche capable de résoudre des instances est LP. Les approches PPC ne résolvent aucune instance en moins de deux heures, alors que certaines instances (ERP 1 et zoo) sont résolues en moins d'une seconde par la PLNE.

Cependant, nous avons constaté que le modèle TB trouve très rapidement la solution optimale. De fait, pour les huit instances considérées, la première solution trouvée par TB est la solution optimale. Les temps mis pour trouver cette solution sont donnés dans la table 5. Ces temps sont inférieurs à 10 secondes pour ERP 1, ERP 2, ERP 3 et zoo, et ils sont inférieurs à 1000 secondes pour ERP 4, tic-tac-toe et vote. Ainsi, TB trouve relativement rapidement la solution optimale mais n'est pas capable de prouver l'optimalité dans la limite de deux heures. Cela provient probablement des heuristiques de choix considérées, qui permettent de guider la recherche vers les bonnes solutions mais ne sont pas efficaces pour prouver l'optimalité.

5 Comparaison de la qualité des solutions en fonction des critères

Dans cette section, nous comparons les clusterings calculés en fonction des critères considérés dans la fonction objectif.

Mesures de performance. Pour évaluer la qualité des clusterings obtenus, nous avons utilisé deux mesures classiques de performance, *i.e.*, la similarité intra-cluster (ICS) et la dissimilarité inter-cluster (ICD). La similarité entre deux transactions est définie par la fonction $s : \mathcal{T} \times \mathcal{T} \rightarrow [0, 1]$ telle que $s(t, t')$ correspond au ratio entre la taille de l'intersection des items de t et t' et la taille de leur union :

$$s(t, t') = \frac{\#\{i \in \mathcal{I} : t\mathcal{R}i \wedge t'\mathcal{R}i\}}{\#\{i \in \mathcal{I} : t\mathcal{R}i \vee t'\mathcal{R}i\}}$$

ICS est la similarité moyenne des paires de transactions appartenant à un même cluster :

$$ICS(C_1, \ldots, C_k) = \frac{1}{2} \sum_{i=1}^{k} (\sum_{t,t' \in C_i} (s(t, t')))$$

Plus ICS est proche de 1, et plus les clusters sont homogènes (*i.e.*, deux transactions à l'intérieur d'un même cluster partagent une grande proportion d'items). Évidemment, cette mesure doit être mise en perspective avec le nombre de transactions dans chaque cluster : une partition où chaque cluster comporte une seule transaction a une valeur ICS égale à 1.

ICD est la dissimilarité moyenne des paires de transaction appartenant à des clusters différents :

$$ICD(C_1, \ldots, C_k) = \sum_{1 \leq i < j \leq k} (\sum_{t \in T_{C_i}, t' \in T_{C_j}} (1 - s(t, t')))$$

Plus ICD est proche de 1, et plus les clusters sont bien séparés (*i.e.*, deux transactions appartenant à des clusters différents partagent peu d'items). Là encore, cette mesure doit être mise en perspective avec le nombre de clusters : une partition comportant un seul cluster a une valeur ICD égale à 1.

La table 6 donne, pour chaque instance, le nombre k de clusters ainsi que les valeurs ICS et ICD des clusterings optimaux selon chacun des trois critères considérés.

Évaluation pour les critères liés à la taille. Considérons tout d'abord les résultats obtenus lorsque le critère à optimiser est lié à la taille des clusters : (1) maximiser la taille minimale ou (3) maximiser la somme des tailles. Dans ce cas, la fusion de deux clusters ne peut que dégrader la qualité de la solution, et donc les solutions optimales ont tendance à avoir un grand nombre de clusters. En pratique, nous observons que pour toutes les instances considérées, le nombre k de clusters dans la solution optimale pour les critères (1) et (3) est égal à $kMax = m - 1$. Autrement dit, tous les clusters ont une seule transaction sauf un qui en a deux. Cela vient du fait que, pour toutes les instances, chaque transaction est un concept formel (autrement dit, pour chaque transaction t_j, il n'existe pas de transaction t_k telle que l'ensemble des items de t_j soit strictement inclus dans l'ensemble des items de t_k, et donc $extent(intent(\{t_j\})) = \{t_j\}$). La conséquence immédiate du fait que les solutions optimales pour les critères (1) et (3) ont $m - 1$ clusters est que ICS est très proche de 1. Nous observons que pour toutes les instances sauf deux (ERP 3 et zoo) les solutions calculées avec les critères (1) et (3) ont des ICS et ICD identiques. Pour ERP 3 et zoo, ICS et ICD sont très proches.

TABLE 6 – Comparaison de la qualité des clusters : chaque ligne donne successivement le nom de l'instance, et la valeur de k, ICS et ICD de la solution optimale pour chacun des trois critères considérés ((1) - maximiser la taille minimale, (2) - maximiser la fréquence minimale, et (3) - maximiser la somme des tailles).

Instance	(1) - taille min.			(2) - fréq. min.			(3) - somme tailles		
	k	ICS	ICD	k	ICS	ICD	k	ICS	ICD
ERP 1	49	**0,9971**	0,4956	2	0,5494	**0,5409**	49	**0,9971**	0,4956
ERP 2	83	**0,9988**	0,3769	2	0,6719	**0,4220**	83	**0,9988**	0,3769
ERP 3	94	0,9966	0,3429	2	0,7242	**0,3876**	94	**0,9993**	0,3311
ERP 4	159	**0,9996**	0,3424	2	0,7072	**0,3945**	159	**0,9996**	0,3424
zoo	58	**0.9980**	0.5724	2	0,4912	**0,5937**	58	0,9945	0,5709
tic tac toe	957	**0,9996**	0,7724	3	0,2919	**0,8042**	957	**0,9996**	0,7724
vote	340	**0,9997**	0,6720	3	0,4279	**0,7277**	340	**0,9997**	0,6720
mushroom	-	-	-	2	0,3793	0,7671	-	-	-

Évaluation pour le critère lié à la fréquence. Considérons maintenant les résultats obtenus lorsque le critère à optimiser est (2) maximiser la fréquence minimale. Dans ce cas, l'éclatement d'un cluster en deux clusters ne peut que dégrader la qualité de la solution, et donc les solutions optimales ont tendance à avoir un petit nombre de clusters. En pratique, toutes les solutions optimales ont 2 clusters, sauf tic-tac-toe et vote qui ont 3 clusters. Dans ce cas, ICS est nettement inférieure à 1 : en moyenne, les transactions d'un même cluster partagent entre 29% (pour tic-tac-toe) et 72% (pour ERP 3) d'items. En contrepartie, ICD est plus importante que pour les critères (1) et (3) qui privilégient la taille : pour le critère (2), la dissimilarité moyenne de transactions de deux clusters différents varie entre 39% (pour ERP 3 et ERP 4) et 80% (pour tic-tac-toe), tandis que pour les critères (1) et (3), elle varie entre 33% ou 34% (pour ERP 3 et ERP 4) et 77% (pour tic-tac-toe).

6 Conclusion

Nous avons proposé une nouvelle approche de clustering conceptuel basée sur la programmation par contraintes où l'ensemble des concepts formels est extrait dans une étape de pré-traitement. L'évaluation expérimentale montre que notre approche est plus efficace en comparaison avec des travaux récents pour des fonctions objectifs maximisant un minimum, obtenant des clusterings de qualité similaire à ceux obtenus en maximisant la somme des tailles des clusters.

Dans cette première série d'expérimentations, nous n'avons pas introduit de contraintes liées à notre application. En particulier, le nombre k de clusters n'est pas contraint et peut prendre n'importe quelle valeur comprise entre 2 et $m-1$. Les premiers résultats montrent que cela ne permet pas d'extraire des clusters à haute valeur ajoutée pour les experts : pour les critères (1) et (3) qui privilégient la taille des clusters, les clusters sont trop nombreux et spécialisés, tandis que pour le critère (2) qui privilégie la fréquence des clusters, ils sont trop peu nombreux et généraux.

Aussi prévoyons-nous dans la suite de nos travaux de rechercher des clusterings plus pertinents en explorant plusieurs pistes. Tout d'abord, nous proposons d'évaluer l'intérêt d'ajouter des contraintes sur la fréquence, la taille et/ou le nombre de clusters, ainsi que d'utiliser de nouveaux critères comme l'aire d'un concept, *i.e.*, le produit de sa taille et de sa fréquence, permettant de considérer en même temps ces deux critères.

Par ailleurs, nous proposons d'appliquer itérativement notre procédure de clustering afin de calculer une hiérarchie de clusterings en adoptant soit une démarche descendante (partitionner progressivement les clusters en partant d'un unique cluster comportant toutes les transactions) ou ascendante (fusionner progressivement les clusters en partant d'une partition comportant un cluster par transaction).

Enfin, nous souhaitons également adapter notre modèle pour faire du bi-clustering étant donné que dans notre cas d'application, un paramétrage peut contenir plusieurs concepts de paramétrage.

Références

[1] Solutions, p.c. : Panorama consulting solutions research report - 2016 erp report, 2016.

[2] M. Munir Ahmad and Ruben Pinedo Cuenca. Critical success factors for erp implementation in smes. *Robot. Comput.-Integr. Manuf.*, 29(3) :104–111, June 2013.

[3] Christian Bessiere, Emmanuel Hebrard, Brahim Hnich, and Toby Walsh. *Disjoint, Partition and Intersection Constraints for Set and Multiset Variables*, pages 138–152. Springer Berlin Heidelberg, 2004.

[4] Thi-Bich-Hanh Dao, Willy Lesaint, and Christel Vrain. Clustering conceptuel et relationnel en programmation par contraintes. In *JFPC 2015*, Bordeaux, France, June 2015.

[5] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.*, 2(2) :139–172, September 1987.

[6] Tias Guns. Declarative pattern mining using constraint programming. *Constraints*, 20(4) :492–493, 2015.

[7] Stephen José Hanson and Malcolm Bauer. Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3(4) :343–372, 1989.

[8] L. Hossain. *Enterprise Resource Planning : Global Opportunities and Challenges : Global Opportunities and Challenges*. IRM Press, 2001.

[9] A. Hotho and G. Stumme. Conceptual clustering of text clusters. In G. Kókai and J. Zeidler, editors, *Proc. Fachgruppentreffen Maschinelles Lernen (FGML 2002)*, pages 37–45, 2002.

[10] Istvan Jonyer, Lawrence B. Holder, and Diane J. Cook. Graph-based hierarchical conceptual clustering in structural databases. In Henry A. Kautz and Bruce W. Porter, editors, *AAAI/IAAI*, page 1078. AAAI Press / The MIT Press, 2000.

[11] Yat Chiu Law and Jimmy H. M. Lee. *Global Constraints for Integer and Set Value Precedence*, pages 362–376. Springer Berlin Heidelberg, 2004.

[12] R.S. Michalski. *Knowledge Acquisition Through Conceptual Clustering : A Theoretical Framework and an Algorithm for Partitioning Data Into Conjunctive Concepts*. Report (University of Illinois at Urbana-Champaign. Dept. of Computer Science). Department of Computer Science, University of Illinois at Urbana-Champaign, 1980.

[13] Brian Neil Mogensen. Goal-oriented conceptual clustering : The classifying attribute approach. *Coordinated Science Laboratory Report no. UILU-ENG-87-2257*, 1987.

[14] Jaideep Motwani, Ram Subramanian, and Pradeep Gopalakrishna. Critical factors for successful erp implementation : Exploratory findings from four case studies. *Comput. Ind.*, 56(6) :529–544, August 2005.

[15] Abdelkader Ouali, Samir Loudni, Yahia Lebbah, Patrice Boizumault, Albrecht Zimmermann, and Lakhdar Loukil. Efficiently finding conceptual clustering models with integer linear programming. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 647–654, 2016.

[16] Ruggero G. Pensa, Céline Robardet, and Jean-François Boulicaut. *A Bi-clustering Framework for Categorical Data*, pages 643–650. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[17] Mike Perkowitz and Oren Etzioni. Towards adaptive web sites : Conceptual framework and case study. *Artificial Intelligence*, 118(1) :245 – 275, 2000.

[18] Lionel Robert, Ashley R. Davis, and Alexander McLeod. Erp configuration : Does situation awareness impact team performance ? *2011 44th Hawaii International Conference on System Sciences (HICSS 2011)*, 00(undefined) :1–8, 2011.

[19] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. *An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases*, pages 16–31. Springer Berlin Heidelberg, 2004.

[20] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.