Efficiently Finding Conceptual Clustering Models with Integer Linear Programming

Abdelkader Ouali[a,b], **Samir Loudni**[b], **Yahia Lebbah**[a], **Patrice Boizumault**[b], **Albrecht Zimmermann**[b] and **Lakhdar Loukil**[a]
(a) Université d'Oran 1 Ahmed Ben Bella, Lab. LITIO, 31000 Oran, Algeria.
(b) University of Caen Normandy, CNRS, UMR 6072 GREYC, 14032 Caen, France.
abdelkader.ouali@unicaen.fr

Abstract

Conceptual clustering combines two long-standing machine learning tasks: the unsupervised grouping of similar instances and their description by symbolic concepts. In this paper, we decouple the problems of finding descriptions and forming clusters by first mining formal concepts (i.e. closed itemsets), and searching for the best k clusters that can be described with those itemsets. Most existing approaches performing the two steps separately are of a heuristic nature and produce results of varying quality. Instead, we address the problem of finding an optimal constrained conceptual clustering by using integer linear programming techniques. Most *other* generic approaches for this problem tend to have problems scaling. Our approach takes advantageous of both techniques, the general framework of integer linear programming, and high-speed specialized approaches of data mining. Experiments performed on `UCI` datasets show that our approach efficiently finds clusterings of consistently high quality.

1 Introduction

Data clustering is one of the foundational data mining tasks. Whether it is for the purpose of "unsupervised classification", data reduction, or anomaly detection, identifying groups of similar data points is an important task. Conceptual clustering [Michalski and Stepp, 1983; Fisher, 1987; Perkowitz and Etzioni, 1999; Pensa *et al.*, 2005] provides an additional aspect: the description of clusters by symbolic concepts, typically the domain of concept learning. Traditional approaches [Michalski and Stepp, 1983; Fisher, 1987] combine the formation of the clusters and of the descriptions, having to trade off the demands of one versus the other. Newer techniques [Perkowitz and Etzioni, 1999; Pensa *et al.*, 2005] have instead chosen to decouple finding the descriptions – either before or after the clustering process – and the clustering step.

Given the large search space to traverse, all those techniques are heuristic, meaning that results are heavily influenced by the initialisation conditions, typically requiring numerous restarts, increasing computational costs. Once those have been performed, an additional question is how to identify the relevant result, as we will illustrate in the experimental section (Section 6).

To address this problem, we combine two exact techniques: we use *closed itemset mining* [Uno *et al.*, 2004] (or *formal concept analysis*) to discover candidates for descriptions, and *integer linear programming* (ILP) to select the best clusters (according to an objective function) that can be described by those descriptions. This results in a single optimal solution to the problem setting.

In ILP, this problem has been introduced as *set partitioning and set covering problems* [Nemhauser and Wolsey, 1988], applied to clustering tasks in [Hansen *et al.*, 1994], and adapted for constrained clustering in [Mueller and Kramer, 2010]. Exact techniques, such as the ones described in this latter work and in [Guns *et al.*, 2013] tend to have problems scaling, and as a side-effect fix the number of clusters to make the problem tractable. In contrast to this, our adoption of closed itemsets cuts down on redundancy compared to other ways of selecting candidate clusters and our use of advanced ILP techniques give our approach additional flexibility.

Our contribution is therefore twofold: we propose an efficient technique for conceptual clustering, based on closed itemsets, that

1. is more flexible in terms of the constraints that can be enforced than existing approaches, and
2. gives more control and stronger guarantees to the user than existing heuristic approaches.

The next section introduces the concepts used in this paper. Section 3 describes how different clustering tasks can be expressed as ILP problems, and Section 4 how to solve them. We discuss related work in Section 5 before demonstrating our technique's performance in Section 6. Section 7 concludes and points towards future research directions.

2 Definitions

In this section we introduce the definitions and concepts that we will use throughout this paper.

2.1 Formal concepts

Let $\mathcal{I}$ be a set of n distinct literals called items, an itemset (or *pattern*) is a non-null subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L}_{\mathcal{I}} = 2^{\mathcal{I}} \backslash \emptyset$. A *transactional dataset* is a

multi-set of m itemsets of $\mathcal{L}_\mathcal{I}$. Each itemset, usually called a *transaction* or object, is a database entry. For instance, Table 1a gives a transactional dataset $\mathcal{T}$ with m=11 transactions $t_1, \ldots, t_{11}$ described by n=8 items A, B, C, D, E, F, G, H.

Let R be a binary relation between the set $\mathcal{T}$ of transactions and the set $\mathcal{I}$ of items s.t. $(t, i) \in R$ if the transaction t contains the item $i : i \in t$. We denote by $\mathcal{R} = (\mathcal{T}, \mathcal{I}, R)$ the tuple formed by these sets of transactions, items and the binary relation. $\mathcal{R}$ is called **formal context** [Ganter and Wille, 1997]. Two operators are defined:

- Let $I \subseteq \mathcal{I}, ext(I) = \{t \in \mathcal{T} | \ \forall i \in I, (t, i) \in R\}$
- Let $T \subseteq \mathcal{T}, int(T) = \{i \in \mathcal{I} | \ \forall t \in T, (t, i) \in R\}$

$ext(I)$ is the set of transactions containing all items in I. $int(T)$ is the set of items contained by all transactions in T. These two operators induce a Galois connection between $2^\mathcal{T}$ and $2^\mathcal{I}$, i.e. $T \subseteq ext(I) \Leftrightarrow I \subseteq int(T)$.

A pair such that $(I = int(T), T = ext(I))$ is called **formal concept**. This definition defines a **closure property** on dataset $\mathcal{T}$, $\texttt{closed}(I) \Leftrightarrow I = int(ext(I))$. An itemset I for which $\texttt{closed}(I) =$ true is called *closed pattern*.

Using $ext(I)$, we can define the *frequency* of a concept: $\texttt{freq}(I) = |ext(I)|$, and its *diversity*: $\texttt{divers}(I) = \sum_{t \in ext(I)} |\{i \in \mathcal{I} \mid (i \notin I) \wedge (i \in t)\}|$. Additionally, we can refer to its *size*: $\texttt{size}(I) = |\{i \mid i \in I\}|$.

2.2 Conceptual clustering

Clustering is the task of assigning the transactions in the data to relatively homogeneous groups. **Conceptual clustering** aims to also provide a distinct **description** for each cluster – the concept characterizing the transactions contained in it.

This problem can be formalized as: "find a set of k clusters, each described by a closed pattern $P_1, P_2, ..., P_k$, covering all transactions without any overlap between clusters". For instance, Table 1b reports the various clusterings for k=3.

An evaluation function f is needed to express the goodness of the clustering. So, conceptual clustering looks for a disjoint set of clusters of $\mathcal{T}$ that optimizes a given criterion of clustering quality. For instance, for dataset $\mathcal{T}$ and k=3, minimizing $f(P_1, ..., P_k) = \sum_{1 \leq i \leq k} \texttt{divers}(P_i)$ provides one clustering s_1, with optimal value 18 (see Table 1b).

2.3 Other clustering settings

Other clustering settings [Berkhin, 2006], e.g. soft clustering, co-clustering, and soft co-clustering, can also be expressed by the relationship between transactions and closed patterns.

a. Co-clustering consists of finding k clusters covering both the set of transactions and the set of items, without any overlap on transactions or on items. For instance, s_1 provides a co-clustering on the transactional dataset $\mathcal{T}$ (see Table 1b).

b. Soft clustering consists of relaxing:

- either the coverage relation: not all transactions are covered, but at least δ_t transactions must be covered, where threshold $\delta_t \leq |\mathcal{T}| (= m)$
- or the non-overlap relation: small overlaps are allowed, but a transaction t cannot occur in more than δ_o clusters, where threshold $\delta_o \geq 1$

c. Soft co-clustering consists of relaxing the coverage or the non-overlap relations for transactions (as for soft clustering), but also the coverage or the non-overlap relations for items:

- not all items are covered, but at least γ_i items must be covered, where threshold $\gamma_i \leq |\mathcal{I}| (= n)$
- small overlaps are allowed, but an item i cannot occur in more than γ_o co-clusters, where threshold $\gamma_o \geq 1$

Example 1 *For δ_t=6, δ_o=1, γ_i=7, γ_o=2, $s_r = [\{A, E\}, \{B, E, G\}, \{C, F, G, H\}]$ is a soft co-clustering since D is the only missing item, items E and G occur twice and $\{t_1, t_7, t_8, t_9, t_{10}\}$ are the uncovered transactions.*

2.4 Integer Linear Programming

Integer Linear Programming (ILP) [Nemhauser and Wolsey, 1988] is one of the most widely used methods for handling optimization problems, due to its rigorousness, flexibility and extensive modeling capability. An ILP program is a linear program with the added restriction that all the variables must be integer. Typically, an ILP model involves: (i) a set of decision variables, (ii) a set of linear constraints, where each constraint requires that a linear function of the decision variables is either equal to, less than, or greater than, a scalar value, and (iii) an objective function that assesses the quality of the solution. Solving an ILP problem consists of finding the best solution w.r.t. the objective function in the set of solutions that satisfy the constraints. Formally, an ILP problem takes the form:

$$\begin{array}{rl} \texttt{Maximize or Minimize} & c^T x \\ \texttt{Subject to} & Ax \ (\leq, =, or \geq) \ b \\ & x_i \in \mathbb{Z}, i = 1..n \end{array} \quad (1)$$

where x represents the vector of decision variables, n is the total number of integer variables, c_j ($1 \leq j \leq n$) are referred to as objective coefficients, A is an $m \times n$ matrix of coefficients, and b is an $m \times 1$ vector of the right-hand-side values of the constraints.

3 ILP models

This section describes ILP models for the different clustering problems introduced in the previous section.

3.1 Conceptual clustering

Let $\mathcal{T}$ be a dataset with m transactions defined on a set of n items $\mathcal{I}$. Let $\mathcal{C}$ be the set of p closed patterns (w.r.t. the frequency measure). Let $a_{t,c}$ be an $m \times p$ binary matrix where $(a_{t,c} = 1)$ iff $c \subseteq t$, i.e., the transaction t belongs to the cluster represented by the closed pattern c.

Hence, the conceptual clustering problem can be modeled as an ILP (see Fig. 1a) using p boolean variables $x_c, (c \in \mathcal{C})$, where $(x_c = 1)$ iff the cluster represented by the closed pattern c belongs to the clustering. Constraints (1) state that each transaction t must be covered by exactly one cluster c. Constraint (2) restricts the number of clusters to k=k_0 clusters. The objective function is defined by associating to each cluster c a value v_c reflecting the interest (maximize) or cost (minimize) of the cluster c, e.g. minimizing each concept's diversity or maximizing their size.

Trans.	Items						
t_1	A	B	D				
t_2	A			E	F		
t_3	A			E		G	
t_4	A			E		G	
t_5		B		E		G	
t_6		B		E		G	
t_7			C	E		G	
t_8			C	E		G	
t_9			C	E			H
t_{10}			C	E			H
t_{11}			C		F	G	H

(a) Transactional dataset $\mathcal{T}$.

Sol.	P_1	P_2	P_3
s_1	{A, B, D}	{C, F, G, H}	{E}
s_2	{B}	{C}	{A, E}
s_3	{A}	{C}	{B, E, G}

(b) Three conceptual clusterings for k=3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
t_1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
t_2	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
t_3	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1
t_4	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1
t_5	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	1
t_6	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	1
t_7	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	1	0	1
t_8	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	1	0	1
t_9	0	0	0	0	0	0	0	1	1	0	1	0	0	1	1	0	0	0
t_{10}	0	0	0	0	0	0	0	1	1	0	1	0	0	1	1	0	0	0
t_{11}	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	1	1

(c) $(a_{t,c})$ matrix associated with dataset $\mathcal{T}$.

Table 1: Running example.

Optimize	$\sum_{c\in\mathcal{C}} v_c \cdot x_c$
Subject to	(1) $\sum_{c\in\mathcal{C}} a_{t,c} \cdot x_c = 1, \quad \forall t \in \mathcal{T}$
	(2) $\sum_{c\in\mathcal{C}} x_c = k_0$
	$x_c \in \{0,1\}, c \in \mathcal{C}$

(a) ILP model for conceptual clustering (M1).

Optimize	$\sum_{c\in\mathcal{C}} v_c \cdot x_c$
Subject to	(1) $\sum_{c\in\mathcal{C}} a_{t,c} \cdot x_c = 1, \quad \forall t \in \mathcal{T}$
	(2) $k = \sum_{c\in\mathcal{C}} x_c$
	(2′) $k_{min} \leq k \leq k_{max}$
	(3) $\sum_{c\in\mathcal{C}} w_{i,c} \cdot x_c = 1, \quad \forall i \in \mathcal{I}$
	$k \in \mathbb{N}, \quad x_c \in \{0,1\}, c \in \mathcal{C}$

(b) ILP model for co-clustering (M2).

Optimize	$\sum_{c\in\mathcal{C}} v_c \cdot x_c$
Subject to	(4) $y_t \leq \sum_{c\in\mathcal{C}} a_{t,c} \cdot x_c \leq \delta_o \cdot y_t, \quad \forall t \in \mathcal{T}$
	(5) $\sum_{t\in\mathcal{T}} y_t \geq \delta_t$
	(6) $z_i \leq \sum_{c\in\mathcal{C}} w_{i,c} \cdot x_c \leq \gamma_o \cdot z_i, \quad \forall i \in \mathcal{I}$
	(7) $\sum_{i\in\mathcal{I}} z_i \geq \gamma_i$
	$k = \sum_{c\in\mathcal{C}} x_c$
	$k_{min} \leq k \leq k_{max}$
	$k \in \mathbb{N},$
	$x_c \in \{0,1\}, \quad c \in \mathcal{C}$
	$y_t \in \{0,1\}, \quad t \in \mathcal{T}$
	$z_i \in \{0,1\}, \quad i \in \mathcal{I}$

(c) ILP model for soft co-clustering (M3).

Figure 1: ILP models for conceptual clustering.

The quality measures we have introduced are related individually to the given patterns. The number of clusters k can also be defined as a variable whose value will be determined by the ILP solver. Constraint (2') illustrates how to control k by specifying a lower bound k_{min} and/or an upper bound k_{max}. Relaxing k removes constraints on the clustering and hence enables finding results from a larger solution space, giving our approach more flexibility (see Section 6.b).

Example 2 *Consider the dataset $\mathcal{T}$ of Table 1a. The $(a_{t,c})$ matrix associated with $\mathcal{T}$ is outlined in Table 1c. If we consider v_c=size(c), then the solution that maximizes the objective function is x_2=x_{12}=x_{15}=1, and all other variables x_c are equal to 0 (so, k=3). This solution corresponds to the clustering s_1 (see Table 1b).*

3.2 Co-clustering

Co-clustering consists of finding k clusters covering both the set of transactions and the set of items, without any overlap on transactions and on items. Let $w_{i,c}$ be an $n \times p$ binary matrix where $(w_{i,c} = 1)$ iff the item i belongs to the closed pattern c. Fig. 1b depicts the ILP model. Constraints (3) state that each item i must be covered by exactly one cluster c.

Example 3 *The optimal solution x_2=x_{12}=x_{15}=1 and all other variables x_c equal to 0, corresponding to s_1 (see Table 1b) is a co-clustering since the associated clusters cover both sets $\mathcal{I}$ and $\mathcal{T}$ without overlap.*

3.3 Soft clustering and soft co-clustering

For soft conceptual clustering (see Section 2.3b), we introduce m boolean variables $y_t, (t \in \mathcal{T})$ such that $(y_t = 1)$ iff transaction t belongs to at least one cluster. Fig. 1c depicts the ILP model. First, as $\sum_{t\in\mathcal{T}} y_t$ indicates the number of covered transactions, Constraint (5) states that at least δ_t transactions must be covered. Second, as a transaction t belongs to $\sum_{c\in\mathcal{C}} a_{t,c}x_c$ clusters, Constraint (4) states that each transaction t must belong to at most δ_o clusters. Note that if y_t=0, then transaction t is not covered and Constraint (4) is satisfied.

We proceed in the same way for soft co-clustering (see Section 2.3c), by introducing n boolean variables $z_i, (i \in \mathcal{I})$ such that $(z_i = 1)$ iff item i belongs to at least one cluster. First, as $\sum_{i\in\mathcal{I}} z_i$ indicates the number of covered items, Constraint (7) states that at least γ_i items must be covered. Second, as an item i belongs to $\sum_{c\in\mathcal{C}} w_{i,c}x_c$ clusters, Constraint (6) states that each item i must belong to at most γ_o clusters. Note that if z_i=0, then item i is not covered and Constraint (6) is satisfied. Example 1 provides an example of soft co-clustering.

Additional constraints can be added in order to avoid undesired clusterings. For instance, one can specify extra constraints on the frequency of closed patterns (e.g. freq$(c) > 1$) which leads to a more balanced clustering for the dataset $\mathcal{T}$, and may avoid outliers receiving their own clusters. In this case, the optimal solution to our running example is x_4=x_7=x_8=1, and all other variables x_c are equal to 0. It corresponds to the conceptual clustering s_3 (see Table 1b).

dataset	#transactions	#items	density(%)	Number of closed itemsets
Soybean	630	50	32	31,759
Primary-tumor	336	31	48	87,230
Lymph	148	68	40	154,220
Vote	435	48	33	227,031
tic-tac-toe	958	27	33	42,711
Mushroom	8124	119	18	221,524
Zoo-1	101	36	44	4,567
Hepatitis	137	68	50	3,788,341
Anneal	812	93	45	1,805,193

Table 2: Dataset characteristics.

4 Preprocessing and solving

We have defined an ILP framework to express various kinds of clusterings. The strength of this framework lies in its flexibility in terms of the constraints that can be enforced. In Section 3, we defined constraints that enforce *global restrictions* on the clusters such as: clusters cover all of the transactions (or a minimum number of them), clusters should be disjoint (or have restricted overlaps), etc. But, in practice, there might be *local constraints* that should be defined on each individual closed pattern such as :

- *minimum* and *maximum* frequency constraints,
- `MustLink`(t_i, t_j) requires that transaction t_i should be in the same cluster as t_j,
- `CannotLink`(t_i, t_j) requires that transaction t_i should not be in the same cluster as t_j.

The more selective local constraints are, the fewer the number of closed patterns for the solving step. Finally, the whole clustering process consists of the following two main steps:

- *Preprocessing step:* compute closed patterns using efficient state-of-the-art extractors, e.g. `LCM`, and filter those violating local constraints. Constraints like minimum frequency and size are directly handled by the extractor. For `CannotLink`(t_i, t_j), any cluster c s.t. t_i and t_j belongs to $ext(c)$ is removed. `MustLink` constraints are handled similarly.
- *Solving step:* solve one of the ILP clustering models detailed in Section 3 using efficient ILP solvers.

5 Related work

Conceptual clustering has been introduced in [Michalski, 1980], and extended in [Michalski and Stepp, 1983; Fisher, 1987]. Numerous approaches have been devised for tackling this problem, e.g. of statistical, syntactic or hierarchical nature [Schalkoff, 2008], with hierarchical conceptual clustering arguably the most prominent one among them [Fisher, 1987; Thompson and Langley, 1991].

Heuristic approaches Several methods have explored the idea of decoupling cluster-formation from finding the conceptual descriptions. Notably, Pensa *et al.* [Pensa *et al.*, 2005] also begin by mining closed (or δ-closed) patterns (itemsets) and their extensions and then perform k-Means clustering on them. Perkowitz and Etzioni [Perkowitz and Etzioni, 1999], reverse the two phases: their *cluster-mining* first uses a clustering technique to form clusters. From the resulting clustering, descriptions are learned by a rule-learning technique. Instances are matched to descriptions, potentially leading to overlapping clusters. Finally, [Geerts *et al.*, 2004] introduced *tiling*. A *tile* is a formal concept, and in that work two settings are considered: i) finding k tiles that cover as many transactions as possible, and ii) finding the minimum number of tiles that cover all transactions. Both are clearly conceptual clustering settings but since tiles can overlap, so can clusters.

CP/SAT for pattern set mining. The CP-paradigm is at the core of generic approaches to deal with k-pattern sets [Khiari *et al.*, 2010; Guns *et al.*, 2013]. These methods allow to model sets of patterns, satisfying properties on the whole set, in a declarative and flexible way, such as conceptual clustering, k-tiling, to name a few, but they look for sets containing a fixed number of patterns and tend to have problems scaling. In [Métivier *et al.*, 2012], the authors formulate the conceptual clustering task as queries in given language. These queries are then translated into SAT clauses and resolved by a SAT solver. However, the proposed framework does not allow to express optimization criteria and has problems scaling.

ILP/CP for distance-based clustering. Distance-based clustering relies upon dissimilarities (or similarities) between numerical attributes. In this line, several declarative approaches have been proposed. In [Dao *et al.*, 2013], a CP approach has been used. The proposed approach supports various kinds of user constraints. In [Babaki *et al.*, 2014], an exact approach, which uses column generation in an ILP setting, is presented. This approach extends the one proposed in [Aloise *et al.*, 2012]. In [Mueller and Kramer, 2010], a constrained clustering approach using ILP is proposed. This approach takes as input a set of possible base clusters and builds a clustering by selecting a suitable subset. The number of clusters k must be given upfront. In contrast, our proposal allows relaxing k by specifying lower/upper bounds. In their experiments, the set of base clusters is generated from frequent patterns. This makes the approach less applicable, because the number of base clusters is huge and selecting a suitable subset is challenging. Our adoption of closed patterns cuts down on redundancy compared to other ways of selecting candidate clusters and makes the solving step more efficient by reducing dramatically the number of candidate clusters.

6 Experiments and Results

The experimental evaluation is designed to address the following questions:

1. How (in terms of CPU-times) does our approach (`CCLP`) compare to the CP approach of Guns et al. (`KPatternSet`) and the approach of Mueller et al.?
2. How do the resulting clusters compare w.r.t. different objective functions?
3. In light of the exact nature of our approach, how do the resulting clusters and their descriptions compare qualitatively with those resulting from existing heuristic approaches that operate in a similar framework? The two approaches with which we compare ourselves are those of Pensa et al. (`CDKMeans`) and Perkowitz et al. (`CM`).

Experimental protocol. Experiments were carried out on the same datasets which were used in [Guns *et al.*, 2013] and available from the UCI repository. Table 2 shows the characteristics of these datasets. All experiments were conducted on AMD Opteron 6282SE with 2.60 GHz of CPU and 512 GB

of RAM. We used LCM to extract the set of all closed patterns and CPLEX $v.12.4$ to solve the different ILP models. For all methods, a time limit of 24 hours has been used.

Experiments have been performed without any local constraints on individual closed patterns. To evaluate the quality of a clustering, we test the coherence of a clustering, measured by the intra-cluster similarity *(ICS)* and the inter-clusters dissimilarity *(ICD)*, both of which should be as large as possible. Given a similarity measure s between two transactions t and t', where $s : \mathcal{T} \times \mathcal{T} \mapsto [0, 1]$, $s(t, t') = \frac{|t \cap t'|}{|t \cup t'|}$

$$ICS(P_1, ..., P_k) = \frac{1}{2} \sum_{1 \leq i \leq k} \Big(\sum_{t,t' \in P_i} s(t, t') \Big)$$

$$ICD(P_1, ..., P_k) = \sum_{1 \leq i < j \leq k} \Big(\sum_{t \in P_i, t' \in P_j} (1 - s(t, t')) \Big)$$

(a) Comparing CCLP with KPatternSet. Figs. 2a and 2c compare the performance of the two methods on model M1 for various values of k on different datasets w.r.t. the first optimization setting (i.e. description size). The CPU-times of CCLP−M1 include those for the preprocessing step. For ($k = 3$), KPatternSet dominates CCLP−M1 except on Soybean, Zoo-1 and Mushroom. But, for ($k \geq 4$), CCLP−M1 outperforms KPatternSet by several orders of magnitude on all datasets. Finally, KPatternSet fails to find a solution within the time limit for ($k > 5$). On Audiology dataset, both approaches were not able to find a solution. This is in part explained by the number of closed patterns (10^6) in comparison to the other datasets (from 10^3 to 10^5). For the second optimization setting (see Fig. 2b), the same observations can be made. Note that on Tic-Tac-Toe no conceptual clustering exists for ($k = 4$).

(b) Impact of relaxing k. To assess the interest of relaxing k, Fig. 2d compares the performance of CCLP−M1 with k relaxed (CCLP−M1−relaxed) against CCLP−M1 in terms of the best value found for k (Col. 2 vs. Col. 5) and CPU-times (Col. 4 vs. Col. 7). Reported in Col. 5 are the best values found for k ($3 \leq k \leq 10$) that maximize the description size. Both settings obtain the same best value for k. Indeed, there is a bias towards a k near k_{max} when using the description size. When comparing the CPU-times, CCLP−M1−relaxed is often *faster*, particularly on the two most difficult datasets Anneal and Hepatitis (speed-up of up to 3.84).

Contrary to Mueller et al., our approach allows to relax k. To compare the two, we implemented the mean of inner cluster similarities (i.e., $s(x, y)$), the objective function used by that work, for fixed k (denoted CCLP−M1−Dist). Fig. 2d compares CCLP−M1−relaxed with CCLP−M1−Dist in terms of the best k found, and CPU-times. Both approaches almost always agree on the best value for k and obtain comparable CPU-times on the first 7 datasets, while on the two difficult datasets CCLP−M1−relaxed is clearly better. Fig. 3d shows that clusterings found by relaxing k are close to those found with fixed k

Finally, in Fig. 2d, one can see that the preprocessing step is negligible as compared to the solving step, except on the two last datasets due to the high number of closed patterns.

(c) Qualitative analysis of clusterings. To assess the quality of clusterings, we have plotted their values according to measures *ICS* and *ICD*. Points (see Fig. 3) represent the optimal clusterings obtained for different values of k on the selected datasets. A specific color is assigned to each dataset. A label, representing the value of k, is associated to some points with the same color. Moreover, we omitted to plot points that are on the Pareto front in Figs. 3b and 3c since they are incomparable. Fig. 3a shows the two optimization criteria of CCLP−M1. Except for the Vote and Tic-Tac-Toe datasets, the diversity measure sacrifices *ICS* to achieve higher *ICD* values. This is indicative of more balanced clusters: the *ICS* is necessarily limited by the number of instances per cluster but the *ICD* increases if there are more instances in other clusters to compare against. The size measure shows the opposite behavior, which is indicative of one (or a few) large clusters, and numerous smaller ones.

To compare with CDKMeans and CM, [1] we first have to identify a good comparison clustering. To that end, (i) we ran each technique 100 times to smooth out the random initialization effects typical for k-Means, (ii) then we grouped the resulting into *equivalence classes*, in which clusterings agree on the composition of *all* clusters, (iii) finally, we chose the largest equivalence class and used its representative (i.e. one of the clusterings in it, since all have the same composition). If there were several equally large equivalence classes, we used a representative from each class.

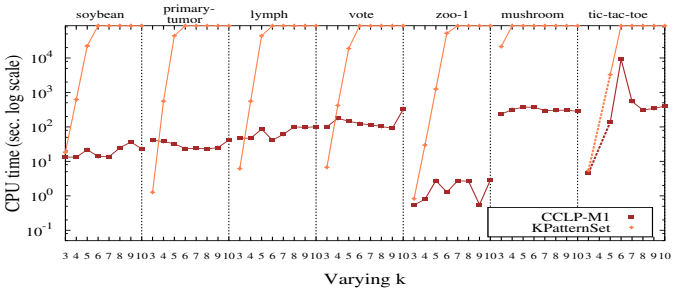
The results show how difficult it can be to control the results of heuristic techniques: the representative clusterings of the CM approach never cover all transactions in the data, and often have overlapping clusters. Fig. 3b shows that the clusterings computed by CM have a qualitative disadvantage compared to the clusterings found by CCLP−M1 (points of CM are always dominated by those of CCLP−M1). For the sake of fair comparison, we run CCLP−M3 with settings that allowed similar amounts of non-coverage. In case that several equally large equivalence classes existed, we chose the clustering that maximized *ICS* and *ICD* and used its coverage as constraint for CCLP−M3. Figure 3c shows that our framework can approximate the results of the heuristic technique.

CDKMeans is even more difficult to control by the user: when using all closed patterns, the algorithm usually assigns all instances to one or two clusters, no matter k. Enforcing frequency constraints improves this situation somewhat even though $|\mathcal{C}| \leq k$ for $k \geq 4$. Additionally, the clusterings occasionally fail to stabilize, resulting in 100 different clusterings. Finally, the particular representation of clusters in CDKMeans can result in clusters that have descriptions yet no transactions assigned to them.

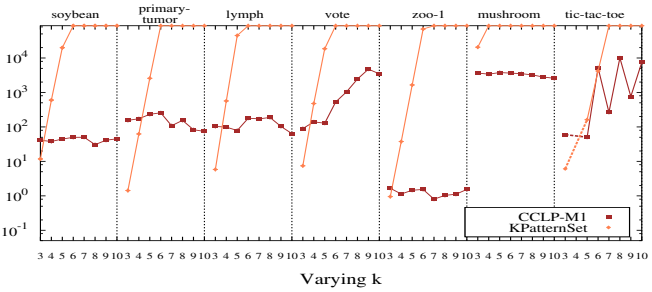
7 Conclusions

We have proposed an efficient approach for conceptual clustering that uses closed itemset mining to discover candidates for descriptions, and ILP to select the best clusters. Closed itemsets cut down on redundancy compared to other ways of selecting candidate clusters and ILP gives our approach more flexibility than exact ones, and stronger guarantees to the user than heuristic ones. We plan to exploit local search techniques to enhance the solving step.

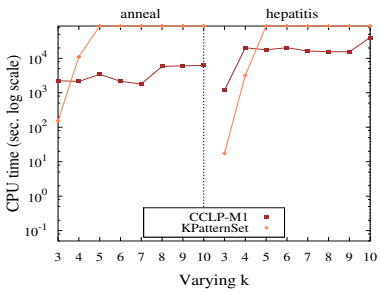
[1] For CM we used SimpleKMeans and JRip in Weka. We reimplemented CDKMeans : http://www.scientific-data-mining.org/software/cdkmeans.zip.



(a) Maximizing description size.



(b) Minimizing pattern diversity.

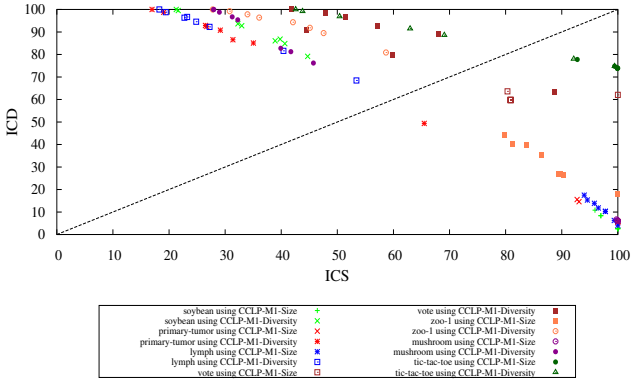


(c) Maximizing the description size.

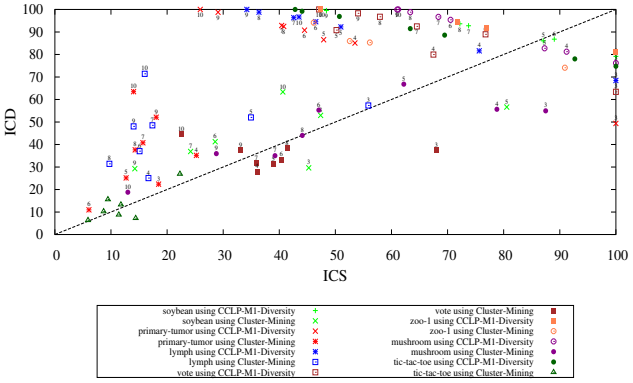
Instance	`CCLP-M1` with k relaxed – maximizing the description size			`CCLP-M1` with k fixed – maximizing the description size			`CCLP-M1-Dist` maximizing the mean of the inner cluster similarities [Mueller and Kramer, 2010]		
	best k	CPU Time (s.)		best k	CPU Time (s.)		best k	CPU Time (s.)	
		(1)	(2)		(1)	(2)		(1)	(2)
Soybean	10	0.74	15.52	10	0.74	21.31	10	0.74	8.52
Primary-tumor	10	1.98	40.62	10	1.98	35.84	8	1.98	35.4
Lymph	10	3.38	25	10	3.38	93.96	10	3.38	149.69
Vote	10	4.42	191.12	10	4.42	304.18	9	4.42	66.21
Zoo-1	10	0.07	6.22	10	0.07	2.73	9	0.07	0.35
Mushroom	10	17.1	154.75	10	17.1	163.67	10	17.1	169.04
Tic-Tac-Toe	10	0.38	420.75	10	0.38	398.86	10	0.38	209.4
Anneal	10	236.85	1,493.38	10	236.85	5,744.03	10	236.85	5,744.03
Hepatitis	10	113.84	20,825.8	10	113.84	40,014.5	10	113.84	37,745.7

(d) Impact of relaxing the number of clusters k ((1) Preprocessing step (2) Solving step).

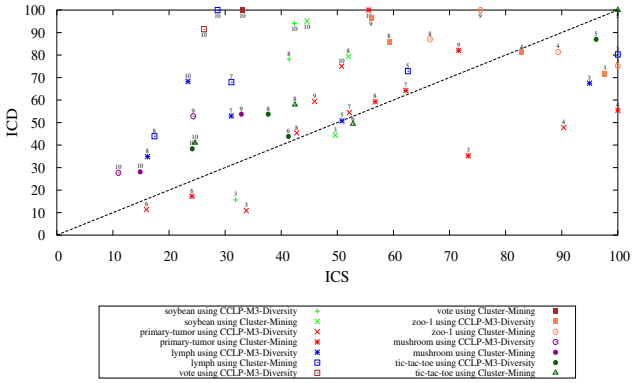
Figure 2: Comparing CPU-times.



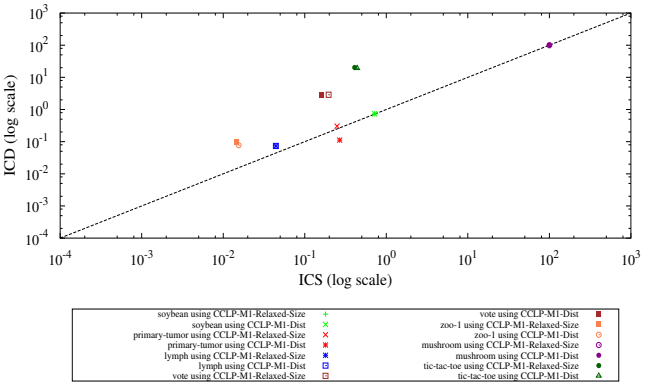
(a) `CCLP-M1`: Diversity vs. Description size.



(b) `CCLP-M1` with Diversity vs. Cluster Mining (`CM`).



(c) `CCLP-M3` with Diversity vs. Cluster Mining (`CM`).



(d) `CCLP-M1-relaxed` with Size vs. `CCLP-M1-Dist`.

Figure 3: Comparing the quality of the resulting clusterings.

Acknowledgements.

This work was partially supported by the Eiffel Excellence program - French Ministry of Foreign Affairs and International Development, REFERENCE: UR / EIFFEL-DOCTORAT 2015 / 840868H.

This work was partly supported by the ANR (French Research National Agency) funded project Hybride ANR-11-BS002-002.

References

[Aloise *et al.*, 2012] D. Aloise, P. Hansen, and L. Liberti. An improved column generation algorithm for minimum sum-of-squares clustering. *Math. Prog.*, 131(1-2):195–220, 2012.

[Babaki *et al.*, 2014] B. Babaki, T. Guns, and S. Nijssen. Constrained clustering using column generation. In *CPAIOR 2014*, pages 438–454, 2014.

[Berkhin, 2006] P. Berkhin. *Grouping Multidimensional Data: Recent Advances in Clustering*, chapter A Survey of Clustering Data Mining Techniques, pages 25–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[Dao *et al.*, 2013] T-B-H. Dao, K-C. Duong, and C. Vrain. A declarative framework for constrained clustering. In *ECML PKDD*, volume 8190 of *LNCS*, pages 419–434, 2013.

[Fisher, 1987] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[Ganter and Wille, 1997] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1997.

[Geerts *et al.*, 2004] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *DS 2004*, pages 278–289, 2004.

[Guns *et al.*, 2013] T. Guns, S. Nijssen, and L. De Raedt. k-pattern set mining under constraints. *IEEE Trans. Knowl. Data Eng.*, 25(2):402–418, 2013.

[Hansen *et al.*, 1994] Pierre Hansen, Brigitte Jaumard, and Eric Sanlaville. *New Approaches in Classification and Data Analysis*, chapter Partitioning Problems in Cluster Analysis: A Review of Mathematical Programming Approaches, pages 228–240. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[Khiari *et al.*, 2010] M. Khiari, P. Boizumault, and B. Crémilleux. Constraint programming for mining n-ary patterns. In *16th CP*, volume 6308 of *LNCS*, pages 552–567, 2010.

[Métivier *et al.*, 2012] J-P. Métivier, P. Boizumault, B. Crémilleux, M. Khiari, and S. Loudni. A constraint language for declarative pattern discovery. In *SAC 2012*, pages 119–125, 2012.

[Michalski and Stepp, 1983] R. S. Michalski and R. E. Stepp. Learning from observation: Conceptual clustering. In *Machine Learning*, pages 331–363. Springer, 1983.

[Michalski, 1980] R. S. Michalski. Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. *Journal of Policy Analysis and Information Systems*, 4(3):219–244, 1980.

[Mueller and Kramer, 2010] M. Mueller and S. Kramer. Integer linear programming models for constrained clustering. In *DS 2010*, pages 159–173, 2010.

[Nemhauser and Wolsey, 1988] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[Pensa *et al.*, 2005] R. G. Pensa, C. Robardet, and J-F. Boulicaut. A bi-clustering framework for categorical data. In *PKDD 2005*, pages 643–650, 2005.

[Perkowitz and Etzioni, 1999] M. Perkowitz and O. Etzioni. Adaptive web sites: Conceptual cluster mining. In *IJCAI 99*, pages 264–269, 1999.

[Schalkoff, 2008] Robert J. Schalkoff. Pattern recognition. In Benjamin W. Wah, editor, *Wiley Encyclopedia of Computer Science and Engineering*. John Wiley & Sons, Inc., 2008.

[Thompson and Langley, 1991] K. Thompson and P. Langley. Concept formation in structured domains. In D. H. Fisher, M. J. Pazzani, and P. Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, pages 127–161. Kaufmann, San Mateo, CA, 1991.

[Uno *et al.*, 2004] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *DS 2004*, pages 16–31, 2004.