Constraint Programming for Multi-criteria Conceptual Clustering

Maxime Chabert, Christine Solnon

Constraint Programming for Multi-criteria Conceptual Clustering

Maxime Chabert[1,2] and Christine Solnon[1]

[1] Université de Lyon, INSA Lyon, LIRIS, F-69622, France
[2] Infologic, France

Abstract. A conceptual clustering is a set of formal concepts (*i.e.*, closed itemsets) that defines a partition of a set of transactions. Finding a conceptual clustering is an $\mathcal{NP}$-complete problem for which Constraint Programming (CP) and Integer Linear Programming (ILP) approaches have been recently proposed. We introduce new CP models to solve this problem: a pure CP model that uses set constraints, and an hybrid model that uses a data mining tool to extract formal concepts in a preprocessing step and then uses CP to select a subset of formal concepts that defines a partition. We compare our new models with recent CP and ILP approaches 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 Resource Planning (ERP) software. We consider two classic criteria to optimize, *i.e.*, the frequency and the size. We show that these criteria lead to extreme solutions with either very few small formal concepts or many large formal concepts, and that compromise clusterings may be obtained by computing the Pareto front of non dominated clusterings.

1 Introduction

Clustering is a non-supervised classification approach which aims at partitioning a set of objects into homogeneous clusters. Conceptual clustering provides, in addition to clusters, a description of clusters by means of formal concepts [15,5]. In this paper, we introduce new Constraint Programming (CP) models to solve this problem, and we evaluate these models on classical academic instances, but also on a new set of instances that comes from a real application case.

Presentation of the applicative context. Enterprise Ressource Planning (ERP) softwares are generic softwares for managing companies. They address many functional goals ranging from commercial management to production or stock management [11]. While the same ERP can be used by many companies, it has to be customized specifically to fit each company needs. This is done thanks to parameters which are used to customize the ERP functionalities to each company depending on his structural and organizational needs. However, the large range of functional goals makes the customization process complex and time consuming. We have studied the customization process of the *Copilote* ERP, developed

by *Infologic* and specialized in food industry management. As pointed out in [21], it appears that most of the time of the customization process is dedicated to the parameterization step: this step basically involves assigning values to parameters in such a way that the ERP fulfills the client needs. The complexity of this step comes from the fact that the ERP has a large number of parameters with strong implicit interactions. Moreover, several studies have shown that this time-consuming parameterization step is less important than human factors (user training, personalized support, for example) [16,1]. Therefore, an important challenge is to reduce the time needed to parameterize an ERP in order to spend more time on human factors.

To reduce the time needed to parameterize an ERP, our goal is to (partially) automate this step. To this aim, we have collected a database of existing parameter settings, corresponding to recent installations of the *Copilote* ERP for 400 clients. We propose to identify relevant groups of parameter settings, and to associate them with functional needs. As many functional needs are common to several clients, these parameter setting groups will be reused when customizing the ERP for a new client with similar needs. To identify relevant parameter setting groups, we propose to partition the database of parameter settings into clusters. As we do not have a relevant measure to evaluate the similarity of different parameter settings, and as most parameters are symbolic ones, we propose to use conceptual clustering to achieve this task: this approach does not assume that there exists a similarity measure, and allows us to describe each cluster by a set of parameter values which are shared by all parameter settings in the cluster.

Contributions and organization of the paper. We introduce the background on conceptual clustering in Section 2. In particular, we describe two declarative approaches which have been recently proposed: [4], that uses Constraint Programming (CP), and [17], that uses Integer Linear Programming (ILP). These declarative approaches are very relevant in our applicative context because they allow us to easily model new constraints or objective functions: a main issue is to find relevant objective functions and constraints to extract setting concepts which make sense for our ERP experts.

In Section 3, we introduce two new CP models to compute optimal conceptual clusterings: the first one is a full CP model; the second one is an hybrid model that uses a dedicated tool to extract formal concepts and uses CP to select the subset of formal concepts that defines an optimal partition.

We experimentally evaluate these models in Section 4. This evaluation is done on some classical academic instances. We also introduce a new benchmark composed of seven instances that have been extracted from our database of parameter settings. In this first evaluation, we mainly consider two objective functions to optimize: the size of the concepts (corresponding to the number of parameters that are common to several clients), and the frequency of the concepts (corresponding to the number of clients that share a common setting). We evaluate scale-up properties of the different approaches when the number of clusters is fixed and when it is not fixed, for each of these objectives separately.

Table 1. Left: Example of transactional dataset with $m = 5$ transactions and $n = 4$ items. Right: Set $\mathcal{F}$ of formal concepts for this dataset.

	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

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

The two objective functions that we consider are complementary and are related to the number of clusters: when maximizing concept sizes, optimal clusterings have many clusters of small frequencies; when maximizing concept frequencies, optimal clusterings have very few clusters of small sizes. In Section 5, we propose to compute the Pareto front of all non-dominated solutions, and we introduce and compare different ways for achieving this with our CP models.

2 Background on Conceptual Clustering

Formal Concepts. Formal Concept Analysis is a way of grouping together objects sharing a same set of attribute values [5]. In this paper, we use the transactional database terminology: objects are called *transactions*, and attribute values are called *items*. More formally, let $\mathcal{T}$ be a set of m transactions (numbered from 1 to m), $\mathcal{I}$ a set of n items (numbered from 1 to n), and $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{I}$ a binary relation that relates transactions to items: $(t, i) \in \mathcal{R}$ denotes the fact that transaction t has item i. We note $itemset(t)$ the set of items associated with t, *i.e.*, $\forall t \in \mathcal{T}, itemset(t) = \{i \in \mathcal{I} : (t, i) \in \mathcal{R}\}$. Given a set E, we note $\mathcal{P}(E)$ the set of all its subsets, and $\#E$ its cardinality.

The *intent* of a subset $T \subseteq \mathcal{T}$ of transactions is the intersection of their itemsets, *i.e.*, $intent(T) = \cap_{t \in T} itemset(t)$. The *extent* of a set $I \subseteq \mathcal{I}$ of items is the set of transactions whose itemsets contain I, *i.e.*, $extent(I) = \{t \in \mathcal{T} : I \subseteq itemset(t)\}$. These two operators induce a Galois connection between $\mathcal{P}(\mathcal{T})$ and $\mathcal{P}(\mathcal{I})$, *i.e.*, $\forall T \subseteq \mathcal{T}, \forall I \subseteq \mathcal{I}, T \subseteq extent(I) \Leftrightarrow I \subseteq intent(T)$. A *formal concept* is a couple $(T, I) \in \mathcal{P}(\mathcal{T}) \times \mathcal{P}(\mathcal{I})$ such that $T = extent(I)$ and $I = intent(T)$. We note $\mathcal{F}$ the set of all formal concepts. The *frequency* of a formal concept (T, I) is the number of transactions, *i.e.*, $freq(T, I) = \#T$, and its *size* is the number of items, *i.e.*, $size(T, I) = \#I$.

For example, we display in table 1 a transactional dataset and its associated set $\mathcal{F}$ of formal concepts. The couple $(\{t_1, t_2\}, \{i_1\})$ is not a formal concept because $intent(\{t_1, t_2\}) = \{i_1, i_4\} \neq \{i_1\}$ and $extent(\{i_1\}) = \{t_1, t_2, t_5\} \neq \{t_1, t_2\}$.

Formal concepts and closed itemset mining. Formal concepts correspond to *closed itemsets* [18] and the set $\mathcal{F}$ may be computed by using algorithms dedicated to the enumeration of frequent closed itemsets, provided that the frequency threshold is set to 1. In particular, LCM [24] is able to extract all formal concepts

of $\mathcal{F}$ in linear time with respect to $\#\mathcal{F}$. As there is usually a huge number of closed itemsets, we may add constraints or optimization criteria to identify relevant concepts. For example, we may search for closed itemsets whose frequency is greater than some threshold and whose size is maximal. We may also combine several criteria, and search for the Pareto front of non dominated formal concepts (where a concept c_1 is dominated by another concept c_2 if c_2 is at least as good as c_1 on all criteria but one, and strictly better on this last criterion). This Pareto front is also called the *skyline* [2].

CP for itemset mining. Using CP to model and solve itemset search problems is a topic which has been widely explored during the last ten years [20,12,8,7]. Indeed, CP allows one to easily model various constraints on the searched itemsets, corresponding to application-dependent constraints for example. These constraints are used to filter the search space during the mining process, and allow CP to be competitive with dedicated mining tools such as LCM. Most recently, [14] introduced a global constraint for extracting frequent closed itemsets. This global constraint enforces domain consistency in polynomial time, and it is quite competitive with LCM: if it is an order slower on basic queries, it is more efficient for complex queries where extra constraints are added. Also, [23] proposed to use CP to extract skyline patterns, *i.e.*, non-dominated patterns according to several criteria: they use a dynamic approach, where constraints are added each time a new solution is found in order to forbid solutions dominated by it.

Conceptual Clustering. Clustering is an unsupervised classification approach the goal of which is to group objects into homogeneous clusters. Conceptual clustering provides, in addition to clusters, a description of clusters by means of formal concepts: each cluster corresponds to a formal concept. More precisely, a conceptual clustering is a set of k formal concepts $\mathcal{C} = \{(T_1, I_1), \ldots, (T_k, I_k)\}$ such that $\{T_1, \ldots, T_k\}$ is a partition of the set $\mathcal{T}$ of transactions.

Different optimization criteria may be considered. In this article, we consider two classical criteria : *minFreq*, to maximize the minimal frequency of a cluster; and *minSize*, to maximize the minimal size of a cluster. For example, let us consider the set $\mathcal{F}$ of formal concepts displayed in table 1. Two examples of clusterings are $P_1 = \{c_1, c_2\}$ and $P_2 = \{c_1, c_7, c_8\}$. According to *minFreq*, the best clustering is P_1 (as the minimal frequency is 2 for P_1 and 1 for P_2). According to *minSize*, both clusterings are equivalent as their minimal size is 1.

The number k of clusters is an important parameter which has a great influence on the size and the frequency of the clusters: small values for k favor clusters with larger frequencies and smaller sizes, whereas large values favor clusters with smaller frequencies and larger sizes.

CP for conceptual clustering. Conceptual clustering is related to closed itemset mining, as each cluster corresponds to a closed itemset. However, the goal is no longer to find closed itemsets that satisfy some constraints or optimize some criteria, but to find a subset of closed itemsets which partitions the set of transactions and optimizes some criteria. Conceptual clustering is a special case of

k-pattern set mining, as introduced in [9]: the conceptual clustering problem is defined by combining a cover and a non-overlapping constraint, and a CP model based on boolean variables is proposed to solve this problem.

[3] describes a CP model for clustering problems where a dissimilarity measure between objects is provided. In this case, the goal is to find a partition of the objects which satisfies some constraints and optimizes an objective function defined by means of this dissimilarity measure. This CP model has been extended to conceptual clustering in [4]. Experimental results reported in [4] show that this model outperforms the binary model of [7]. This model assumes that the number of clusters is defined by a constant k. There is an integer variable G_t for each transaction $t \in \mathcal{T}$: G_t represents the cluster of t and its domain is $D(G_t) = [1, k]$. Symmetries (due to the fact that cluster numbers may be swapped) are broken by adding a $precede(G, [1, k])$ constraint [13]. Each cluster is enforced to have at least one transaction by the constraint: $atLeast(1, G, k)$. For each cluster $c \in [1, k]$, a set variable $Intent_c$ represents the intent of the set of transactions in c, *i.e.*, the intersection of their itemsets. Its domain is the set of all possible itemsets, *i.e.*, $D(Intent_c) = \mathcal{P}(\mathcal{I})$. The extent constraint is expressed by: $\forall c \in [1, k], \forall t \in \mathcal{T}, G_t = c \Leftrightarrow Intent_c \subseteq itemset(t)$. It is implemented thanks to $k \times m$ reified constraints (with $m = \#\mathcal{T}$). The intent constraint is expressed by: $\forall c \in [1, k], Intent_c = \cap_{t \in \mathcal{T}, G_t = c} itemset(t)$. It is implemented thanks to k constraints, and each of these k constraints needs n reified domain constraints to build the set of all transactions in cluster c, and a set element global constraint to select the corresponding itemsets and intersect them. An objective variable is introduced. Depending on the optimization criterion, this variable is constrained to be equal either to the minimal cardinality of all $Intent_c$ variables (*minSize*), or the minimal number of G variables assigned to a same cluster thanks to $atLeast(obj, G, c)$ constraints (*minFreq*).

The model proposed in [4] assumes that the number of clusters is fixed to a constant value k. It may easily be extended to the case where this number is not known, by introducing an integer variable k, whose domain is bounded between 2 and $m - 1$. However, performance is degraded when k is not fixed.

ILP for Conceptual Clustering. [17] proposes to compute conceptual clusterings by combining two exact techniques: in a first step, a dedicated closed itemset mining tool (*i.e.*, LCM [24]) is used to compute the set $\mathcal{F}$ of all formal concepts and, in a second step, ILP is used to select a subset of $\mathcal{F}$ that is a partition of the set $\mathcal{T}$ of transactions and that optimizes some given criterion. More precisely, for each formal concept $f \in \mathcal{F}$, there is a binary variable x_f such that $x_f = 1$ iff f is selected. The subset of selected formal concepts is constrained to define a partition of $\mathcal{T}$ by posting the constraint: $\forall t \in \mathcal{T}, \sum_{f \in \mathcal{F}} a_{tf} x_f = 1$, where $a_{tf} = 1$ if transaction t belongs to the extension of concept f, and 0 otherwise. Contrary to the CP approaches of [7,3], the number of clusters is not fixed and it is a variable k which is constrained to be equal to the number of selected concepts by posting the constraint: $k = \sum_{f \in \mathcal{F}} x_f$. In [17], the goal is to maximize the sum of the sizes of the selected concepts. Therefore, the objective function to maximize is: $\sum_{f \in \mathcal{F}} v_f x_f$ where v_f is the size of the concept f. If the case is not

explicitly discussed in [17], we may easily extend this ILP model to maximize the minimal size (resp. frequency) of the selected concepts: we introduce a variable v_{min} and enforce this variable to be smaller than or equal to the size (resp. the frequency) of the selected concepts by adding the constraint $\forall t \in T, v_{min} \leq v_f x_f + M(1-x_f)$, where M is a positive constant greater than the largest possible size (resp. frequency), and v_f is the size (resp. frequency) of the concept f.

3 New CP models

In this section, we introduce two new CP models for computing optimal conceptual clusterings. The first model (described in Section 3.1) may be seen as an improvement of the CP model of [3]. The second model (described in Section 3.2) follows the two step approach of [17]: the first step is exactly the same; the second step uses CP to select formal concepts. These models are experimentally evaluated and compared with the approaches of [3] and [17] in Section 4.

For both models, we do not assume that the number of clusters is fixed: k is a variable whose domain is $[k_{min}, k_{max}]$, where k_{min} and k_{max} are two given bounds such that $2 \leq k_{min} \leq k_{max} < m$.

3.1 New full CP model

Like the CP model of [3], we use G_c integer variables to model clusters. However, we associate the $Intent$ set variables to transactions instead of associating them to clusters. This simplifies the propagation of the intent constraint. Another reason for associating $Intent$ set variables to transactions instead of clusters is that, when k is strictly lower than k_{max}, each $Intent_c$ set variable such that $c > k$ should be empty. This would imply to use reification to compute the minimal intent size, as we must not consider $Intent_c$ variables such that $c > k$ (otherwise, the minimal size would be equal to 0 whenever $k < k_{max}$). Also, we introduce new set variables to explicitly model extents and these set variables are associated with transactions to ease the computation of the minimal frequency. Finally, we introduce redundant set variables which model concept extents and are associated with clusters: these variables are used to add a redundant partition constraint which improves the solution process.

More formally, we use the following variables:

- an integer variable k (with $D(k) = [k_{min}, k_{max}]$), which represents the number of clusters;
- for each transaction $t \in \mathcal{T}$:
  - an integer variable G_t (with $D(G_t) = [1, kMax]$), which represents the cluster of t;
  - a set variable $Intent_t$ (with $D(Intent_t) = \mathcal{P}(itemset(t))$), which represents the set of items in the intent of the cluster of t;
  - a set variable $Extent_t$ (with $D(Extent_t) = \mathcal{P}(\mathcal{T})$), which represents the set of transactions in the extent of the cluster of t;

– for each cluster $c \in [1, k_{max}]$, a set variable $ExtentCluster_c$ (with $D(ExtentCluster_c) = \mathcal{P}(\mathcal{T})$), which represents the set of transactions in c;
– two integer variables $minFreq$ (with $D(minFreq) = [1, m-1]$) and $minSize$ (with $D(minSize) = [1, n-1]$), which represent the minimal frequency and size, respectively.

As proposed in [4,3], we break symmetries (due to the fact that clusters may be swapped) by posting the constraint: $precede(G, [1, k_{max}])$.

We relate $extent_t$ and $extentCluster_c$ variables by posting the constraint: $\forall t \in \mathcal{T}, Extent[t] = ExtentCluster[G_t]$, and we relate $ExtentCluster_c$ and G_t variables by posting the constraint

$$\forall t \in \mathcal{T}, \forall c \in [1, k_{max}], t \in ExtentCluster_c \Leftrightarrow G_t = c$$

We add a redundant partition constraint that enforces $extent$ to be a partition of $\mathcal{T}$: $partition(ExtentCluster, \mathcal{T})$. This constraint is redundant because each transaction is already enforced to belong to exactly one cluster by G variables. However, its propagation both reduces the search space and the CPU time.

We reify $m(m-1)/2$ equality constraints between G variables to ensure that two transactions are in a same cluster iff they have the same intent, and this intent is included in their itemsets: $\forall \{t_1, t_2\} \subseteq \mathcal{T}$

$$(G_{t_1} = G_{t_2}) \Leftrightarrow (Intent_{t_1} = Intent_{t_2}) \Leftrightarrow (Intent_{t_1} \subseteq itemSet(t_2))$$

This constraint ensures the extent property as any transaction t_1 such that $itemset(t1) \supseteq Intent_{t2}$ is constrained to be in the same cluster as t_2. However, this constraint only partially ensures the intent property: for each transaction t, it ensures $Intent_t \subseteq \cap_{t' \in \mathcal{T}, G_t = G_{t'}} itemset(t')$ whereas the intent property requires that $Intent_t$ is equal to the itemset intersection. However, given any solution that satisfies the constraint $Intent_t \subseteq \cap_{t' \in \mathcal{T}, G_t = G_{t'}} itemset(t')$, we can easily transform it to ensure that it fully satisfies the intent property by adding to $Intent_t$ every item $i \in (\cap_{t' \in \mathcal{T}, G_t = G_{t'}} itemset(t')) \setminus Intent_t$. Hence, each time a solution is found, for each cluster c, we compute its actual intent by intersecting the intersection of all its transaction itemsets. This ensures that each cluster actually is a formal concept, and therefore this ensures correction. Completeness is ensured by the fact that our constraint is a relaxation of the initial constraint.

Finally, we compute the minimal size and frequency by posting the constraints: $minSize = \min_{t \in \mathcal{T}} \#Intent_t$ and $minFreq = \min_{t \in \mathcal{T}} \#Extent_t$.

The search strategy depends on the objective function: if the goal is to maximize $minFreq$, then we first assign k to its lower values (as a smaller number of clusters usually leads to concepts with larger frequencies), whereas if the goal is to maximize $minSize$, then we first assign k to its higher values (as a larger number of clusters usually leads to concepts with larger sizes).

3.2 New hybrid model

This model solves the problem in two steps as in [17]: in a first step, we extract the set $\mathcal{F}$ of all formal concepts with a dedicated tool (LCM), and in a second step we use CP to select the subset of $\mathcal{F}$ forming an optimal clustering.

We have designed and compared several CP models for this second step. In particular, we have designed a model that associates a set variable $Extent_c$ with each cluster c (such that $Extent_c$ contains all transactions in the extent of c), and then post a *partition* global constraint on these variables to ensure that they form a partition of $\mathcal{T}$. This model is always outperformed by the model described below.

Our CP model for the second step uses the following variables:

- an integer variable k (with $D(k) = [k_{min}, k_{max}]$), which represents the number of clusters (*i.e.*, the number of selected concepts);
- a set variable P (with $D(P) = \mathcal{P}(\mathcal{F})$), which represents the set of selected formal concepts that define an optimal clustering;
- for each transaction $t \in \mathcal{T}$, an integer variable $Concept_t$ (with $D(Concept_t) = \{f \in \mathcal{F} \mid t \in extent(f)\}$), which represents the concept that contains t in its extent (each transaction must belong to exactly one selected concept);
- two integer variables $minFreq$ (with $D(minFreq) = [1, m-1]$) and $minSize$ (with $D(minSize) = [1, n-1]$), which represent the minimal frequency and size, respectively.

To ensure that $Concept_t$ belongs to P, for each transaction $t \in \mathcal{T}$, we post the constraint: $\forall t \in \mathcal{T}, member(Concept_t, P)$.

To ensure that the selected concepts define a partition of $\mathcal{T}$, we ensure that each transaction t is contained in the extent of exactly one selected formal concept. To this aim, we compute, for each transaction t, the set $CF(t)$ of all the concepts of $\mathcal{F}$ that contain t in their extent, *i.e.*,

$$\forall t \in \mathcal{T}, CF(t) = \{f \in \mathcal{F} : t \in extent(f)\}$$

and we ensure that the set variable P contains exactly one element of $CF(t)$ by posting the constraint: $\forall t \in \mathcal{T}, \#(CF(t)) \cap P) = 1$.

Finally, we compute the minimal size and frequency by posting the constraints: $minSize = \min_{t \in \mathcal{T}} \#intent(C_t)$ and $minFreq = \min_{t \in \mathcal{T}} \#extent(C_t)$.

The number of clusters of the solution is constrained in two different ways according to the objective function:

- If the goal is to maximize $minFreq$, optimal solutions often have a small number of clusters and, in this case, we ensure that k is equal to the number of distinct values contained in C by posting the constraint: $nValue(C, k)$.
- If the goal is to maximize $minSize$, optimal solutions often have a large number of clusters and, in this case, we ensure that k is equal to the cardinality of P by posting the constraint $\#P = k$.

The search strategy also depends on the objective function. The idea is to first select concepts with large sizes (resp. frequency) when the goal is to maximize $minSize$ (resp. $minFreq$). To this aim, we sort formal concepts by decreasing size (resp. frequency), and use this order as value ordering heuristic for C_t. We use a *First fail* strategy to select the variable with the smallest domain as next variable.

Table 2. Test instances: each row gives the number of transactions ($\#\mathcal{T}$), the number of items ($\#\mathcal{I}$), the density (d), the number of formal concepts ($\#\mathcal{F}$), and the CPU time (in seconds) spent by LCM to extract all formal concepts.

Instance	$\#\ \mathcal{T}$	$\#\ \mathcal{I}$	$d(\%)$	$\#\ \mathcal{F}$	Time
ERP1	50	27	48	1 580	0.01
ERP2	47	47	58	8 1337	0.03
ERP3	75	36	51	10 835	0.03
ERP4	84	42	45	14 305	0.05
ERP5	94	53	51	63 633	0.28
ERP6	95	61	48	71 918	0.45
ERP7	160	66	45	728 537	5.31

Instance	$\#\ \mathcal{T}$	$\#\ \mathcal{I}$	$d(\%)$	$\#\ \mathcal{F}$	Time
zoo	59	36	44	4 567	0.01
vote	341	48	34	227 031	0.54
tic-tac-toe	958	27	33	42 711	0.05
soybean	303	116	29	817 534	6.7

Furthermore, when the goal is to maximize $minFreq$, we use the *ObjectiveStrategy* proposed by Choco [19], which performs a dichotomous branching over the domain of $minFreq$.

4 Experimental comparison for single objective problems

We compare our new models with the CP model of [3] and the ILP model of [17] for computing conceptual clusterings that optimize a single objective.

Experimental protocol. All experiments were conducted on Intel(R) Core(TM) i7-6700 with 3.40GHz of CPU and 65GB of RAM. The approach of [4] (called *FullCP1*) is implemented in Gecode v4.3. The approach of [17] (called ILP) uses LCM to extract formal concepts and Cplex v12.7 to solve the selection problem. Our CP model described in Section 3.1 (called *FullCP2*) is implemented in Choco v.4.0.3 [19]. Our hybrid approach described in Section 3.2 (called *HybridCP*) uses LCM v5.3 to extract formal concepts and Choco v.4.0.3 to solve the selection problem. We have limited the CPU time of each run to 1000 seconds.

Test instances. We consider four classical machine learning instances, coming from the UCI database: zoo, vote, tic-tac-toe, and soybean. We also introduce seven new instances (called ERPi, with $i \in [1, 7]$) that have been extracted from our ERP database. Our ERP database contains 400 parameter settings: each setting corresponds to the customization of the ERP for a different client, and specifies the values of almost 450 different parameters. Each parameter can only take a finite number of values, and most of them are symbolic attributes. For each parameter/value couple, we have created a boolean item (set to 1 iff the parameter is assigned to the value in the setting). We have split this database into smaller ones by focusing on different groups of parameters, thus obtaining seven instances of various sizes[3]. All instances are described in Table 2.

[3] These instances are available on `http://liris.cnrs.fr/csolnon/ERP.html`.

Table 3. Times when the goal is to optimize *minFreq* (upper part) and *minSize* (lower part): each line gives the time of the four approaches when k is fixed to 2, 3, and 4, respectively, and when k is not fixed (N). '-' is reported when time exceeds 1000s.

	ILP				FullCP1				FullCP2				HybridCP			
	k=2	k=3	k=4	N	k=2	k=3	k=4	N	k=2	k=3	k=4	N	k=2	k=3	k=4	N
Objective = Maximize *minFreq*																
ERP1	0.2	0.9	1.0	0.8	**0.0**	**0.0**	**0.3**	**0.2**	0.2	0.7	4.3	0.3	0.2	0.9	1.4	0.3
ERP2	1.5	2.7	**2.3**	1.0	**0.0**	0.4	4.8	0.5	0.1	**0.2**	2.8	**0.1**	4.4	1.5	4.6	0.3
ERP3	1.5	2.5	3.2	1.7	**0.0**	**0.3**	20.0	2.4	0.2	1.5	**1.6**	**0.3**	9.2	24.7	2.4	0.6
ERP4	7.5	15.0	**20.9**	13.6	**0.0**	**0.3**	36.6	1.2	0.3	2.8	37.9	**0.4**	1.4	100.6	153.1	0.8
ERP5	12.5	18.3	**83.7**	18.3	**0.0**	**1.4**	773.6	125.3	0.5	5.0	91.9	**1.5**	172.2	634.4	-	10.6
ERP6	52.6	145.8	339.6	143.3	**0.0**	10.3	302.7	51.7	0.5	**2.7**	**101.1**	**1.1**	8.6	-	-	8.0
ERP7	-	-	-	-	**0.0**	82.9	-	973.4	2.8	**26.8**	**742.9**	**5.0**	-	-	-	-
zoo	1.0	2.2	3.0	1.5	**0.0**	**0.0**	**0.8**	**0.1**	0.2	0.2	4.5	0.3	0.5	0.6	1.0	0.2
vote	40.6	-	-	55.2	**0.0**	**2.0**	292.6	-	1.6	19.2	370.5	33.1	17.8	150.0	**95.4**	**20.8**
tic-tac-toe	61.3	80.6	-	718.6	**0.2**	**0.3**	106.0	-	32.5	75.9	-	179.7	10.9	25.2	-	**33.3**
soybean	-	-	-	-	**0.1**	160.1	-	-	1.4	**7.9**	**166.0**	-	63.7	980.2	-	-
Objective = Maximize *minSize*																
ERP1	0.2	0.3	**0.3**	0.4	**0.0**	**0.1**	1.0	0.2	0.3	0.7	2.5	0.2	0.2	0.4	1.6	**0.1**
ERP2	1.7	1.6	1.6	0.8	**0.0**	0.5	19.9	-	0.1	**0.2**	**0.8**	**0.0**	4.6	17.7	7.2	0.1
ERP3	1.6	1.6	**1.7**	1.2	**0.0**	**0.6**	252.9	-	0.3	2.0	7.0	**0.1**	9.6	42.5	61.6	0.2
ERP4	7.5	8.3	**7.2**	18.3	**0.0**	**0.8**	184.8	2.1	0.5	4.6	34.4	0.5	22.0	103.4	329.5	**0.3**
ERP5	13.1	21.1	**40.6**	12.5	**0.0**	**2.2**	-	-	0.6	6.1	58.4	**0.3**	-	-	-	1.5
ERP6	63.4	93.3	648.3	-	**0.0**	14.2	**9.6**	7.2	0.8	**7.5**	54.2	**0.5**	645.0	-	-	1.9
ERP7	-	-	-	-	**0.0**	191.1	-	47.2	4.4	**69.2**	**682.5**	**2.3**	-	-	-	39.5
zoo	1.1	0.9	**1.2**	2.0	**0.0**	**0.0**	1.4	0.5	0.2	1.7	7.7	0.1	0.7	6.8	9.4	**0.1**
vote	40.8	243.5	249.7	-	**0.0**	**3.9**	969.4	-	3.3	12.2	**191.8**	20.4	16.2	69.0	-	**17.2**
tic-tac-toe	60.7	80.4	-	254.5	**0.4**	**0.3**	**105.6**	-	33.2	54.1	-	-	10.9	25.9	-	**18.7**
soybean	-	-	-	-	**0.0**	145.7	-	-	2.5	**7.1**	**93.3**	**22.2**	93.4	460.6	-	342.4

Computation of $\mathcal{F}$ with LCM. Table 2 displays the time spent by LCM to extract all formal concepts, for each instance. This time is proportional with the size of $\mathcal{F}$, as the complexity of LCM is linear with respect to $\#\mathcal{F}$. CPU time is smaller than one second for all instances but two, and it is smaller than seven seconds for the instance that has the largest number of formal concepts (soybean).

Comparison of scale-up properties of the different approaches. Table 3 displays the times for computing optimal clusterings when the goal is to maximize *minFreq* (upper part) or *minSize* (lower part). We report times when k is fixed to 2, 3, and 4, respectively, and then when k is not fixed (with $k_{min} = 2$ and $k_{max} = m - 1$). Times displayed for ILP and HybridCP both include the time spent by LCM to extract all formal concepts.

For all approaches and all instances, time increases when increasing k from 2 to 4. FullCP1 and FullCP2 are more efficient when $k = 2$ than when k is not fixed, and they are more efficient when k is not fixed than when $k = 4$. HybridCP approach needs more time to solve the problem when $k = 2$ than when k is not fixed for all instances but three (vote, tic-tac-toe and soybean) whereas ILP approach needs more time only for the smallest ERP instances.

When $k = 2$, the best approach is FullCP1, which is able to solve all instances in less than 0.1 second (except tic-tac-toe). However, when increasing k from 2

Table 4. Experimental comparison of frequencies, sizes, and number of clusters: each line displays the optimal values of *minFreq* and *minSize* when k is set to 2, 3, and 4, and when k is not fixed (N), followed by the value of k in the optimal solution (in brackets). Finally, it displays the optimal values of *minFreq* and *minSize* when maximizing *minFreq* and breaking ties with *minSize* (Freq+Size), and when maximizing *minSize* and breaking ties with *minFreq* (Size+Freq). In this case, k is not fixed and its value in the optimal solution is displayed in brackets.

	Maximize *minFreq*					Maximize *minSize*					Freq+Size			Size+Freq		
k	2	3	4	N	(k)	2	3	4	N	(k)	Freq	Size	(k)	Freq	Size	(k)
ERP1	21	14	11	21	(2)	4	5	6	12	(49)	21	4	(2)	1	12	(49)
ERP2	21	15	11	21	(2)	6	11	13	16	(42)	21	6	(2)	2	16	(8)
ERP3	31	22	18	31	(2)	3	4	6	12	(59)	31	2	(2)	1	12	(59)
ERP4	42	27	18	42	(2)	3	6	8	18	(83)	42	3	(2)	1	18	(83)
ERP5	41	30	22	41	(2)	3	6	7	16	(79)	41	2	(2)	1	16	(79)
ERP6	42	31	22	42	(2)	8	11	9	28	(94)	42	8	(2)	1	28	(94)
ERP7	70	50	35	70	(2)	5	8	11	29	(159)	70	5	(2)	1	29	(159)
zoo	28	19	14	28	(2)	2	3	5	15	(58)	28	2	(2)	1	15	(58)
vote	102	34	34	102	(2)	1	1	1	15	(317)	102	1	(2)	1	5	(317)
tic-tac-toe		250		250	(3)		1		7	(957)	250	1	(3)	-	-	-
soybean		2	5	-	-		1	1	6	(302)	-	-	-	1	6	(302)

to 3, times of FullCP1 are strongly increased (up to 191 seconds for ERP7 with *minSize*) and FullCP1 is outperformed by FullCP2 for 8 instances. When further increasing k to 4, FullCP2 becomes the only approach able to solve all instances but tic-tac-toe, though ILP is able to solve 8 instances quicker.

When k is not fixed, the best performing approaches are fullCP2 (which is able to solve all instances but soybean for *minFreq*) and HybridCP (which is able to solve all instances for *minSize*).

Maximization of the sum of sizes. In [17], the objective function to maximize is the sum of the sizes of the selected concepts, and the proposed ILP model scales well for this objective: when k is not fixed, the time needed to find the optimal solution is 0.1, 0.4, 0.7, 1.7, 5.9, 14.0, and 183.2 for ERP1 to ERP7, respectively, and it is 0.3, 51.9, 32.0, and 120.0 for zoo, vote, tic-tac-toe, and soybean, respectively. Hence, ILP is more efficient for maximizing the sum of the sizes than for maximizing *minSize*. None of the CP models considered here scales well when the goal is to maximize the sum of the sizes, and they are far slower than ILP in this case: they usually very quickly find the optimal solution, but they are not efficient to prove optimality. However, we noticed that the optimal solutions found with the two criteria sum of sizes and *minSize* are very similar (and often equal). Indeed, when maximizing the minimal size, we also tend to maximize the size of all concepts.

Comparison of frequencies, sizes, and number of clusters of optimal solutions. Table 4 displays the values of *minFreq*, *minSize*, and k for the optimal solutions

(tic-tac-toe does not have clusterings when $k \in \{2, 4\}$, and soybean does not have clusterings when $k = 2$). It shows us that when k is fixed, the optimal values of *minFreq* and *minSize* often greatly vary when modifying the value of k. For example, let us consider instance ERP4: *minFreq* decreases from 42 to 27 and 18 (resp. *minSize* increases from 3 to 6 and 8) when k is increased from 2 to 3 and 4. From an applicative point of view, finding the relevant value for k is not straightforward. When k is not fixed, we obtain extreme solutions: when the goal is to maximize *minFreq*, there are only 2 clusters (except for tic-tac-toe, as there is no solution with $k = 2$), and when the goal is to maximize *minSize*, there are $m - 1$ clusters for all instances but 4 (ERP2, ERP3, ERP5 and vote), whereas for the remaining instances the value of k is rather high.

Table 4 also displays the values of *minFreq*, *minSize*, and k when optimizing the two criteria in a lexicographic order and not fixing k. Let us call Freq+Size the solution that maximizes *MinFreq* while breaking ties with *MinSize*, and Size+Freq the solution that maximizes *MinSize* while breaking ties with *MinFreq*. These solutions correspond to very different situations: for Freq+Size, k is always equal to 2 (except for tic-tac-toe) and *minSize* is rather low (ranging between 1 and 8); for Size+Freq, k is equal to $m - 1$ for 6 instances, and rather large for the other instances, while *minFreq* is always equal to 1, except for ERP2. From an applicative point of view, these solutions are not very interesting, and we need to find better compromises between size and frequency.

5 Multi-criteria optimization

As the two optimization criteria tend to produce extreme solutions which are not very meaningful for our application, we propose to compute the Pareto front of non dominated solutions. A clustering C_1 is dominated by another clustering C_2 if the size and the frequency of C_1 are smaller than or equal to the size and the frequency of C_2. Non dominated solutions correspond to different compromises between the two criteria. The two extrema of the Pareto front are the solutions called Size+Freq and Freq+Size in the previous Section. In Section 5.1, we experimentally evaluate the efficiency of our two CP models for computing these extrema. Then, in Section 5.2, we propose and compare different approaches for computing the whole Pareto front.

5.1 Computation of extrema solutions

Table 5 displays the time spent by FullCP2 and HybridCP to compute the two extrema solutions of the Pareto front: Size+Freq is computed by first maximizing *MinSize*, fixing *MinSize* to its optimal value, and then maximizing *MinFreq*; Freq+Size is obtained by first maximizing *MinFreq*, fixing *MinFreq* to its optimal value, and then maximizing *MinSize*.

For Freq+Size, FullCP2 outperforms HybridCP on 6 instances and it is able to solve all instances except soybean while HybridCP is not able to solve ERP7, vote and soybean. However, for Size+Freq, FullCP2 is not able to solve 5 instances, while HybridCP is able to solve all instances but 2.

Table 5. Times needed to compute Size+Freq (left part) and Freq+Size (rightpart) for FullCP2 and HybridCP. For each solution, we first give the time needed to optimize the first criterion (1st), then the time needed to optimize the second criterion (2nd), and finally the total time ('-' if total time exceeds 1000s).

Instance	Size+Freq						Freq+Size					
	FullCP2			HybridCP			FullCP2			HybridCP		
	1st	2nd	Total	1st	2nd	Total	1st	2nd	Total	1st	2nd	Total
ERP1	0.1	0.2	**0.3**	0.3	0.3	0.6	0.4	0.2	0.7	0.2	0.1	**0.4**
ERP2	-	-	-	0.1	0.3	**0.4**	0.2	0.1	**0.3**	0.3	0.3	0.6
ERP3	-	-	-	0.2	0.4	**0.6**	0.4	0.3	**0.7**	0.7	0.3	1.0
ERP4	0.4	6.9	7.3	0.6	1.5	**2.1**	0.5	0.5	1.0	0.8	0.1	**0.9**
ERP5	-	-	-	1.6	34.7	**36.3**	1.1	0.6	**1.6**	10.6	13.0	23.5
ERP6	0.5	1.4	**1.9**	4.0	94.2	98.2	1.6	0.7	**2.3**	8.0	59.8	67.9
ERP7	3.7	975.1	**978.8**	-	-	-	6.3	2.9	**9.2**	-	-	-
zoo	0.1	1.0	1.1	0.2	0.2	**0.4**	0.4	0.3	0.7	0.3	0.0	**0.3**
vote	20.0	6.2	**26.2**	18.3	648.7	667.0	26.5	13.5	**40.0**	-	-	-
tic-tac-toe	-	-	-	-	-	-	230.9	152.1	383.0	31.6	19.1	**50.7**
soybean	-	-	-	325.0	342.1	**667.1**	-	-	-	-	-	-

5.2 Computation of the Pareto front

[3] describes a CP approach to compute non dominated bi-criteria clusterings by iteratively solving single criterion optimization problems while alternating between the two criteria. [6] describes a more dynamic CP approach to compute Pareto front: the idea is to search for all solutions, and dynamically add a constraint each time a new solution is found to prevent the search from computing solutions that are dominated by it. This idea has been improved in [22,10]. We have experimentally compared these two approaches, and found that the dynamic approach of [6] is more efficient than the static approach of [3] for our problem. Hence, we only consider this approach in this section. It proceeds as follows: we build an initial model as described in Section 3, and ask the solver to search for all solutions. Each time a solution *sol* is found, we dynamically post the constraint $(minFreq > f) \vee (minSize > s)$ where f and s are the values of *minFreq* and *minSize* in *sol*, and go on the search for all solutions. The search stops when there is no more non-dominated solutions.

We have evaluated this dynamic approach with FullCP2 and HybridCP. FullCP2 is able to solve ERP1 in ten minutes, but fails to solve all other instances within a time limit of two hours. HybridCP is much more efficient, and is able to solve 6 instances within this time limit. Hence, we only consider HybridCP in our experiments.

We have compared different variants of this dynamic approach. For all variants, we use the two extrema solutions to reduce the search space in an *a priori* way. More precisely, let $f_{\text{Freq+Size}}$ and $s_{\text{Freq+Size}}$ (resp. $f_{\text{Size+Freq}}$ and $s_{\text{Size+Freq}}$) be the value of *minFreq* and *minSize* in the solution Size+Freq (resp. Freq+Size).

We set the domain of *minFreq* to $[f_{\text{Size+Freq}} + 1, f_{\text{Freq+Size}} - 1]$ and the domain of *minSize* to $[s_{\text{Freq+Size}} + 1, s_{\text{Size+Freq}} - 1]$.

The first two variants correspond to the dynamic approach described below, with different search heuristics: $freq_{seq}$ (resp. $size_{seq}$) uses the search heuristics dedicated to the *minFreq* (resp. *minSize*) objective as described in Section 3.2. These two variants find complementary solutions at the beginning of the search process: $freq_{seq}$ first finds clusterings with large frequencies, whereas $size_{seq}$ first finds clusterings with large sizes. Hence, the variant $freqSize_{par}$ takes advantage of this complementarity and launches the two variants in two parallel threads which communicate their solutions to update the non-dominated area: each time a solution is found by one thread, it dynamically adds constraints to filter the solutions dominated by this solution, and it also checks whether the other thread has found new solutions and dynamically adds constraints if ever.

The variant $freq{+}D_{seq}$ (resp. $size{+}D_{seq}$) decomposes the problem into two subproblems by separating the domain of *minFreq* (resp. *minSize*) in two equal parts. The two subproblems are solved sequentially. We first solve the subproblem corresponding to the upper part of the domain, as no solution of this problem can be dominated by a solution of the other subproblem. Then, we solve the second subproblem while preventing it from computing solutions that are dominated by the solutions of the first subproblem by adding constraints. The variants $freq{+}D_{par}$ and $size{+}D_{par}$ are similar to $freq{+}D_{seq}$ and $size{+}D_{seq}$: the only difference is that they solve the two subproblems in two parallel threads. In this case, only one subproblem (the one with the upper part of the domain) communicates its solutions to the other thread.

Table 6 compares times of these different variants on 6 instances with a time limit of 2 hours (all other instances cannot be solved in less than 2 hours). For all instances, $size_{seq}$ is much more efficient than $freq_{seq}$. Launching these two approaches in two parallel threads does not pay off: $freqSize_{par}$ is faster than $size_{seq}$ for only two instances. This may come from the fact that $freq_{seq}$ is really not efficient compared to $size_{seq}$. Decomposing the problem into two subproblems appears to be a better idea, even when solving the two subproblems sequentially, for the *freq*-based variants: $freq{+}D_{seq}$ is always much faster than $freq_{seq}$. However, $size{+}D_{seq}$ is faster than $size_{seq}$ for two instances only. Finally, solving the two subproblems on two parallel threads always pays-off for the *freq*-based variants, whereas it degrades the solving time for 4 instances for the *size*-based variants. Figure 1 displays the Pareto fronts for ERP4 and ERP5.

6 Conclusion

We have introduced new CP models for computing optimal conceptual clusterings. These models are able to quickly find solutions that maximize either the minimal size or the minimal frequency, even when the number of clusters is not fixed. Computing the Pareto front for these two criteria is a more challenging problem, and our CP models are able to solve this problem in less than two hours for six instances only. Further work will mainly aim at improving this.

Table 6. Times to find all non-dominated solutions with different variants ('-' if time exceeds 2 hours). $\frac{seq}{par}$ gives the speed-up between sequential and parallel variants (for $freqSize_{par}$, we consider the best sequential time).

	ERP1		zoo		ERP4		ERP3		ERP2		ERP5	
	time	$\frac{seq}{par}$	time	$\frac{seq}{par}$	time	$\frac{seq}{par}$	time	$\frac{seq}{par}$	time	$\frac{seq}{par}$	time	$\frac{seq}{par}$
$freq_{seq}$	2.3		24.4		6048.7		145.7		41.7		5542.5	
$size_{seq}$	1.4		8.6		211.8		42.3		**11.1**		**873.9**	
$freqSize_{par}$	1.4	1.0	7.0	1.2	216.5	1.0	36.0	1.2	12.7	0.9	1029.9	0.8
$freq{+}D_{seq}$	**0.8**		4.7		236.8		15.0		12.6		1579.6	
$freq{+}D_{par}$	**0.8**	1.0	**4.5**	1.0	210.0	1.1	**10.2**	1.5	11.8	1.1	1006.9	1.6
$size{+}D_{seq}$	**0.8**		10.6		264.3		24.8		12.4		2912.9	
$size{+}D_{par}$	1.1	0.7	15.0	0.7	**166.0**	1.6	59.1	0.4	24.4	0.5	2131.6	1.4

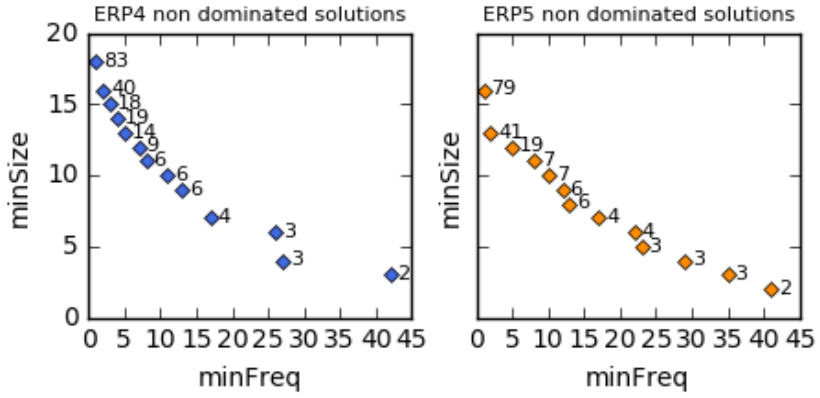

Fig. 1. Pareto front of ERP4 and ERP5: each point (x, y) corresponds to a non-dominated solution with x=*minFreq* and y=*minSize*. The number k of clusters of the solution is displayed close to the point.

In particular, we plan to combine different decompositions to obtain more than two subproblems, *e.g.*, both decompose the domains of *minFreq* and *minSize* to obtain four subproblems that may be solved in four parallel threads. We also plan to evaluate scale-up properties of ILP for this problem, and combine ILP with CP if we observe complementary performance. Finally, we plan to evaluate the interest of combining our CP model with the propagation algorithm of [14].

Acknowledgments. We thank Jean-Guillaume Fages and Charles Prud'homme for enriching discussions on Choco, and authors of [4] for sending us their Gecode code.

References

1. 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.
2. Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *Proceedings of the 17th IEEE International Conference on Data Engineering*, pages 421–430, 2001.
3. Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Constrained Clustering by Constraint Programming. *Artificial Intelligence*, pages –, June 2015.
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. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1997.
6. Marco Gavanelli. An algorithm for multi-criteria optimization in csps. In *Proceedings of the 15th European Conference on Artificial Intelligence*, ECAI'02, pages 136–140, Amsterdam, The Netherlands, The Netherlands, 2002. IOS Press.
7. Tias Guns. Declarative pattern mining using constraint programming. *Constraints*, 20(4):492–493, 2015.
8. Tias Guns, Siegfried Nijssen, and Luc De Raedt. Itemset mining: A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983, 2011.
9. Tias Guns, Siegfried Nijssen, and Luc De Raedt. k-pattern set mining under constraints. *IEEE Trans. Knowl. Data Eng.*, 25(2):402–418, 2013.
10. Renaud Hartert and Pierre Schaus. A support-based algorithm for the bi-objective pareto constraint. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 2674–2679. AAAI Press, 2014.
11. L. Hossain. *Enterprise Resource Planning: Global Opportunities and Challenges: Global Opportunities and Challenges*. IRM Press, 2001.
12. Mehdi Khiari, Patrice Boizumault, and Bruno Crémilleux. Constraint programming for mining n-ary patterns. In *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, pages 552–567, 2010.
13. Yat Chiu Law and Jimmy H. M. Lee. *Global Constraints for Integer and Set Value Precedence*, pages 362–376. Springer Berlin Heidelberg, 2004.
14. Nadjib Lazaar, Yahia Lebbah, Samir Loudni, Mehdi Maamar, Valentin Lemière, Christian Bessiere, and Patrice Boizumault. A global constraint for closed frequent pattern mining. In *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, pages 333–349, 2016.
15. 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.
16. 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.
17. 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.
18. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Database Theory - ICDT '99, 7th International Conference*, pages 398–416, 1999.
19. Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016.
20. 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, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 204–212, 2008.
21. 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.
22. Pierre Schaus and Renaud Hartert. *Multi-Objective Large Neighborhood Search*, pages 611–627. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
23. Willy Ugarte, Patrice Boizumault, Bruno Crémilleux, Alban Lepailleur, Samir Loudni, Marc Plantevit, Chedy Raïssi, and Arnaud Soulet. Skypattern mining: From pattern condensed representations to dynamic constraint satisfaction problems. *Artif. Intell.*, 244:48–69, 2017.
24. 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.