Efficiently Mining Closed Interval Patterns with Constraint Programming

Djawad Bekkoucha[1], Abdelkader Ouali[1], Patrice Boizumault[1], and Bruno Crémilleux[1]

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, Caen, FRANCE
firstname.lastname@unicaen.fr

Abstract. Constraint programming (CP) has become increasingly prevalent in recent years for performing pattern mining tasks, particularly on binary datasets. While numerous CP models have been designed for mining on binary data, there does not exist any model designed for mining on numerical datasets. Therefore these kinds of datasets need to be pre-processed to fit the existing methods. Afterward a post-processing is also required to recover the patterns into a numerical format. This paper presents two CP approaches for mining closed interval patterns directly from numerical data. Our proposed models seamlessly execute pattern mining tasks without any loss of information or the need for pre- or post-processing steps. Experiments conducted on different numerical datasets demonstrate the effectiveness of our proposed CP models compared to other methods.

Keywords: Constraint Programming · Pattern Mining · Numerical Data.

1 Introduction

Pattern mining is a crucial task in data mining. The constraint-based pattern mining task [15] embraces a wide range of queries and methods aimed at extracting characterized patterns from data. These patterns can be interpreted by domain experts or serve as inputs for downstream tasks such as classification or clustering [5].

Datasets commonly encountered in many application domains often comprise a variety of value types, including binary, discrete, numerical or mixed values. There are a plethora of pattern mining methods on binary data, sequences or graphs [15] but few methods address numerical [8, 18] or mixed values [4]. A simple approach to cope with numerical data is to reuse existing methods by first converting data into a binary representation [6]. However, it is well-known that the binarization process often leads to a loss of information. Considering the example of interval patterns defined by numerical values, MININTCHANGE [8] is a method aiming to mine such patterns directly from numerical data. However it is devoted to closed interval patterns and cannot deal with other data mining tasks.

The aim of our work is to propose a comprehensive declarative framework to efficiently tackle numerical data. Our work is guided by two principles. First, we keep the whole original information expressed by the numerical data. Second, we choose to set our approach in a declarative paradigm, the Constraint Programming (CP) framework. The declarative paradigm enables to specify pattern mining tasks in an easy way by

using general constraint primitives. In the last few years, many declarative constraint-based methods were proposed for mining tasks on binary data or sequences [5, 9, 16], to the best of our knowledge, so far, there is no declarative method tackling numerical datasets. The core of our contribution is a declarative method to mine the complete set of closed interval patterns from numerical data without any discretization process. A closed pattern is essential in data mining tasks as it captures the maximum amount of similarity within a dataset. These patterns have crucial properties and are widely used in data mining applications. Moreover the whole set of interval patterns can be regenerated following the principle of the pattern condensed representations [2]. Our CP models go beyond interval patterns by combining primitives to write queries addressing complex mining tasks on numerical data. As an illustration, we combine the no overlapping between patterns and the cover of the set of the mined patterns to perform the conceptual clustering task.

Our contributions are the following. We define two CP models to mine the complete set of closed interval patterns. The first model, called CP4CIP, is based on reified constraints and offers a general encoding of the closure relation to mine closed patterns. In CP, global constraints can capture hidden relations between a set of variables to improve the efficiency. We design a global constraint, called GC4CIP, using new specific filtering rules, and we give a second CP model enhancing the mining efficiency. We provide an extensive empirical evaluation comparing the efficiency of our CP models with respect to other declarative methods and the ad hoc method MININTCHANGE. Finally, we illustrate the interest of the declarativity by using GC4CIP in a scenario of data processing chain to find conceptual clustering from numerical data.

Section 2 introduces notations and basic concepts. Section 3 is devoted to related work. Section 4 presents our first CP model based on reified constraints. Section 5 and Section 6 depicts our second CP model based on global constraints. Section 7 provides an extensive empirical evaluation on benchmark datasets and Section 8 concludes.

2 Preliminaries

2.1 Interval Pattern Mining

Numerical dataset. A *numerical dataset* $\mathcal{N}$ is defined by a set of objects $\mathcal{G}$ where each object is described by a set of attributes $\mathcal{M}$. Each attribute $m \in \mathcal{M}$ has a range $\mathcal{N}_m$ which is a finite set containing all the possible values of m in $\mathcal{N}$. An object $g \in \mathcal{G}$ is defined by a vector of numerical values $< v_{g,m} >_{m \in \{1,\ldots,|\mathcal{M}|\}}$. A dataset where the values of all attributes are binary $\mathcal{N}_m = \{0, 1\}, \forall m \in \mathcal{M}$, is a special case of a numerical dataset and referred as a *binary dataset*.

Example 1. Table 1 shows a running example of a numerical dataset containing 5 objects $\mathcal{G} = \{g_1, g_2, g_3, g_4, g_5\}$, each object is described by 3 attributes $\mathcal{M} = \{m_1, m_2, m_3\}$.

Closed Interval Pattern. Patterns in numerical datasets can be represented in many ways, we use the notion of Interval Pattern [8] which is defined as a vector of intervals $\mathcal{V} = \langle [\underline{w_m}, \overline{w_m}] \rangle_{\forall m \in \mathcal{M}}$, where $\underline{w_m}, \overline{w_m} \in \mathcal{N}_m$. Each dimension of the vector $\mathcal{V}$ corresponds to an attribute following a canonical order on the set of attributes $\mathcal{M}$. We denote

	m_1	m_2	m_3
g_1	2	8	130
g_2	4	12	102
g_3	3	7	91
g_4	2	9	101
g_5	6	12	110

Table 1. A running example of a numerical dataset $\mathcal{N}$

$\mathcal{B}[g] = \langle [v_{g,m}, v_{g,m}] \rangle_{m \in \{1,...,|\mathcal{M}|\}}$ as the vector of intervals corresponding to an object identified by g. An object g is an occurrence of the interval pattern $\mathcal{V}$ if each interval in the vector $\mathcal{B}[g]$ is included in the interval of $\mathcal{V}$, i.e. $\mathcal{B}[g] \sqsubseteq \mathcal{V} \iff [v_{g,m}, v_{g,m}] \subseteq [\underline{w}_m, \overline{w}_m], \forall m \in \{1,...,|\mathcal{M}|\}$. The cover of $\mathcal{V}$ in $\mathcal{N}$ is the set of objects $g \in \mathcal{G}$ occurring in $\mathcal{V}$, i.e. $cover(\mathcal{V}) = \{g \in \mathcal{G} \mid \mathcal{B}[g] \sqsubseteq \mathcal{V}\}$.

Example 2. From Table 1, $\mathcal{V} = \langle [3,4], [7,12], [91,130] \rangle$ is an interval pattern covering the objects $\{g_2, g_3\}$. $\mathcal{B}[g_2]$= $\langle [4,4], [12,12], [102,102] \rangle$ is the vector of intervals identified by the object g_2 and an occurrence of $\mathcal{V}$.

The frequency of $\mathcal{V}$ is the cardinal of its cover, i.e. $freq(\mathcal{V}) = |cover(\mathcal{V})|$. Given a minimum frequency threshold θ, the interval pattern $\mathcal{V}$ is frequent if and only if $freq(\mathcal{V}) \geq \theta$. A description of a subset of objects $G \subseteq \mathcal{G}$ is an interval pattern $\mathcal{V}$ where for each $g \in G$, g is an occurrence of $\mathcal{V}$, i.e. $desc(G) =< [a_m, b_m] >_{m \in \{1,...,|\mathcal{M}|\}}$ such that $a_m = min(\{v_{g,m} \mid g \in G\})$ and $b_m = max(\{v_{g,m} \mid g \in G\})$.

Exact pattern condensed representations (such as the *closed patterns* [19]) enable to reduce the large number of patterns extracted from the datasets without loss of information [2]. The key idea of pattern condensed representations is to take advantage of the redundancy of a collection of patterns to construct a concise representation of the patterns instead of mining all patterns. A closed interval pattern (CIP) is defined by the closure of an interval pattern that is the vector of intervals of its cover (i.e. $close(\mathcal{V}) \iff desc(cover(\mathcal{V})) = \mathcal{V}$).

Example 3. From Table 1, the set of objects $\{g_2, g_3\}$ is described by the interval pattern $\mathcal{V} = desc(\{g_2, g_3\}) = \langle [3,4], [7,12], [91,102] \rangle$. The interval pattern $\mathcal{V} = \langle$ [3,4], [7,12], [91,102]$\rangle$ is closed since $desc(\{g_2, g_3\}) = \langle [3,4],$ [7,12], [91,102]$\rangle$ and the $cover(\langle [3,4],$ [7,12], [91,102]$\rangle) = \{g_2, g_3\}$.

2.2 Problem statement

Our goal is to discover all frequent closed interval patterns (FCIP). More formally, given a numerical dataset $\mathcal{N}$ and a minimum frequency threshold θ, the closed frequent interval pattern mining problem is the problem of finding *all* interval patterns $\mathcal{V}$ such that $freq(\mathcal{V}) \geq \theta$ and $close(\mathcal{V})$.

There are a few things one should note about this statement. First, the search space contains $\sum_{k=1}^{|\mathcal{G}|} \binom{|\mathcal{G}|}{k}$ candidates. This size is gigantic and a naive search that consists of enumerating and testing the frequency of all interval pattern candidates is not practical.

Second, mining FCIP can be solved by the following process: transform the numerical data using Interordinal Scaling (IS) technique to get binary data, mine closed itemsets from the binary data and post-processing closed itemsets to obtain closed interval patterns [8]. Results are equivalent because IS preserve all the information of the original data by creating pairs of binary attributes of the following form: $m \leq v_{g,m}$ and $m \geq v_{g,m}, \forall m \in \mathcal{M}, g \in \mathcal{G}$. Each element of these pairs is then used as a binary attribute on the set of objects. The value of the attribute on each object is set to 1 if the condition holds, otherwise the value is set to 0. However, this approach produces a large dataset having $\sum_{m \in \mathcal{M}} 2|\mathcal{N}_m|$ items. Moreover, the post-processing step is highly expensive. For each itemset found, it needs to determine the interval of each attribute by calculating the minimum and the maximum values that are present in the itemset. The time complexity of the post-processing is in the worst case $\mathcal{O}(\mathcal{C} \cdot |\mathcal{M}| \cdot \mathcal{U})$ where $\mathcal{C}$ is the total number of mined closed itemsets and $\mathcal{U}$ is the number of distinct values for each attribute.

Finally, using the declarative paradigm easily enables us to combine CIP with other constraints such as the overlapping or the cover of the set of returned patterns [9, 7, 3]. We illustrate the use of these constraints with our CP models in Section 7.3.

2.3 Constraint programming

A CSP consists of a set of variables $X = \{x_1, \ldots, x_n\}$, a set of domains $\mathcal{D}$ mapping each variable $x_i \in X$ to a finite set of possible values $\mathcal{D}(x_i)$, and a set of constraints $\mathcal{C}$ on X. A constraint $c \in \mathcal{C}$ is a relation specifying the allowed combinations of values for its variables $X(c)$. An assignment on a set $Y \subseteq X$ of variables is a mapping from variables in Y to values in their domains. A solution is an assignment on X satisfying all the constraints. CP solvers typically use backtracking tree search to explore the search space of partial assignments and attempts to extend them to consistent ones with the objective of finding solutions. The main technique used to speed-up the search is the constraint propagation by a filtering algorithm. Each constraint filtering should remove as many variable domain values as possible by enforcing local consistency properties like *domain* or *bound consistency*. Global constraints are constraints capturing a relation between a fixed number of variables. These constraints provide the solver with a better view of the structure of the problem. Dedicated filtering algorithms are designed to achieve better time complexity.

3 Related Work

Mining patterns in numerical data started with quantitative association rule mining [18]. A lot of work is discussed in [17]. Many of them are based on a natural approach where each numerical attribute is discretized according to some interest functions, e.g. support, class values. Then patterns are mined from the discretized data. This family of approaches leads to a loss of information. More recently, in the field of subgroup discovery, a very common kind of patterns in modern pattern mining, Nguyen et al. [14] creates a discretization process with the goal to maximize the average quality of the patterns. [12] provides a thorough comparison of existing methods to deal with numerical attributes in subgroups. By considering the notion of closed interval patterns, OSMIND [13] finds optimal subgroups according to an interestingness measure in purely

numerical data. Our work takes advantage of the principle of the closed interval patterns and it is not limited to subgroups.

Approaches based on Minimum Description Length are used for discovering useful patterns and returning a set of non-redundant overlapping patterns with well-defined boundaries [11, 20]. In order to design relevant intervals on the fly based on numerical data, MININTCHANGE [8] introduces a framework that enumerates all closed interval patterns starting with the largest one, then explores the search space through minimal changes on the interval bounds. The principle has been reused to search for patterns corresponding to convex polygons [1] but the technique is limited to two dimensions. All these methods are dedicated to specific patterns and require to rewrite algorithms when the problem at hands changes.

There are many declarative methods for constraint-based mining tasks under declarative frameworks for binary data or sequences [5, 9, 16]. CP-based approaches have been proposed to mine closed itemsets in a binary context as CP4IM [16] by using reified constraints or the global constraint closedPattern [10]. However, to the best of our knowledge, there is no declarative method to discover patterns directly from numerical data without requiring pre and post processing steps.

4 First Model using Reified Constraints

This section starts by presenting the variables used to model interval patterns and their associated domains. Then, we describe our first model named CP4CIP.

4.1 Variables and Domains

The bounds of an interval pattern are modelled by introducing two variables $\underline{x}_m$, $\overline{x}_m$ for each attribute $m \in \mathcal{M}$. These variables represent respectively the lower bound and the upper bound. The domains of these variables is the set of values $\mathcal{N}_m$ in the dataset, i.e. $\mathcal{D}(\underline{x}_m) = \mathcal{D}(\overline{x}_m) = \mathcal{N}_m$.

Additionally, we introduce another set of variables, denoted by Y, to capture the coverage of an object by an interval pattern. The variable $y_g \in Y$ is associated to the object $g \in \mathcal{G}$ and has a binary domain, i.e. $\mathcal{D}(y_g) = \{0, 1\}$. The variable y_g takes the value 1 if and only if the object g is covered by the candidate interval pattern. A FCIP is found by setting the variables $\underline{x}_m$ and $\overline{x}_m$ to a value in $\mathcal{N}_m$ and y_g to 0 or 1.

Example 4. From Table 1, the interval pattern $\langle [3,4], [7,12], [91,102] \rangle$ is modelled by the following assignment $\{\underline{x}_1 = 3,\ \overline{x}_1 = 4,\ \underline{x}_2 = 7,\ \overline{x}_2 = 12,\ \underline{x}_3 = 91,\ \overline{x}_3 = 102, y_1 = 0, y_2 = 1, y_3 = 1, y_4 = 1, y_5 = 0\}$.

4.2 Reified Constraints

Coverage constraints. An object is covered by a FCIP iff all values of its attributes are found in the intervals. To formulate the cover of a FCIP, we introduce in our model Boolean variables $B_{g,m}$, for each value m and each object g in the database such that:

$$B_{g,m} = 1 \iff \underline{x}_m \leq v_{g,m} \leq \overline{x}_m,\ \forall m \in \mathcal{M}, \forall g \in \mathcal{G} \tag{1}$$

The following constraint exploits the variables $B_{g,m}$ to enforce the cover on each object. An object $g \in \mathcal{G}$ is covered iff the value of each attribute m for the object g is bounded by the interval $[\underline{x}_m, \overline{x}_m]$. We can thus process the frequency of the interval pattern by summing the y_g variables such that the sum must be greater or equal than a minimum support θ (i.e. $\sum_{g\in\mathcal{G}} \ yg \ \geq \theta$).

$$y_g = 1 \iff \sum_{m\in\mathcal{M}} B_{g,m} = |\mathcal{M}|, \ \forall g \in \mathcal{G} \tag{2}$$

Proof. Let $\mathcal{V}$ be a candidate interval pattern. The variable y_g models the cover of object $g \in \mathcal{G}$ by $\mathcal{V}$. The cover of $\mathcal{V}$ is given by the following: $y_g = 1 \iff \underline{x}_m \leq v_{g,m} \leq \overline{x}_m, \forall \ m \in \mathcal{M}$. Since, $B_{g,m} = 1 \iff \underline{x}_m \leq v_{g,m} \leq \overline{x}_m$. So, $y_g = 1 \iff B_{g,m} = 1, \forall m \in \mathcal{M}$. It follows that $y_g = 1 \iff \sum_{m\in\mathcal{M}} B_{g,m} = |\mathcal{M}|$.

Closure constraints. The closure requires that each interval associated to each attribute should contain all the values of the covered objects while each value of an uncovered object should be outside of the interval. Let $\mathcal{N}_m^{\uparrow}$ (resp. $\mathcal{N}_m^{\downarrow}$) be the maximum (resp. the minimum) value over the objects on the attribute m, the closure relation can be expressed in our model by introducing the new variables $\underline{H}_{g,m}$ and $\overline{H}_{g,m}$, where $\mathcal{D}(\underline{H}_{g,m}) = \{v_{g,m}\} \cup \{\mathcal{N}_m^{\uparrow} + 1\}$, and $\mathcal{D}(\overline{H}_{g,m}) = \{v_{g,m}\} \cup \{\mathcal{N}_m^{\downarrow} - 1\}$. The upper value $\{\mathcal{N}_m^{\uparrow} + 1\}$ (resp. lower value $\{\mathcal{N}_m^{\downarrow} - 1\}$) is added in the domain of $\underline{H}_{g,m}$ (resp. $\overline{H}_{g,m}$) in order to avoid selecting the minimum (resp. maximum) on the uncovered objects.

Lower bound closure. To capture the minimum value of covered objects, we use for each attribute $m \in \mathcal{M}$ a minimum constraint on the set variables $\{\underline{H}_{g,m}, \forall g \in \mathcal{G}\}$. The variables $\underline{H}_{g,m}$ of uncovered objects have value greater than all the values in the data. Thus, the minimum cannot be selected on uncovered objects.

$$\forall g \in \mathcal{G}, m \in \mathcal{M}, y_g = 1 \implies \underline{H}_{g,m} = v_{g,m} \quad (3) \qquad \forall g \in \mathcal{G}, m \in \mathcal{M}, y_g = 1 \implies \overline{H}_{g,m} = v_{g,m} \quad (6)$$

$$\forall g \in \mathcal{G}, m \in \mathcal{M}, y_g = 0 \implies \underline{H}_{g,m} = \mathcal{N}_m^{\uparrow} + 1 \quad (4) \qquad \forall g \in \mathcal{G}, m \in \mathcal{M}, y_g = 0 \implies \overline{H}_{g,m} = \mathcal{N}_m^{\downarrow} - 1 \quad (7)$$

$$\forall m \in \mathcal{M}, \underline{x}_m = min(\underline{H}_{1,m}, \underline{H}_{2,m}, ..., \underline{H}_{|\mathcal{G}|,m}) \quad (5) \qquad \forall m \in \mathcal{M}, \overline{x}_m = max(\overline{H}_{1,m}, \overline{H}_{2,m}, ..., \overline{H}_{|\mathcal{G}|,m}) \quad (8)$$

Upper bound closure. Similarly, to find the maximum value on the covered objects, we use for each attribute m a maximum constraint on the set of variables $\{\overline{H}_{g,m}, \forall g \in \mathcal{G}\}$. The variables $\overline{H}_{g,m}$ of uncovered objects have value smaller than all the values in the data. Thus, the maximum cannot be selected on uncovered objects.

Example 5. Table 2 shows the values taken by the closure variables $\underline{H}_{g,m}$ (left) and $\overline{H}_{g,m}$ (right) to find the closed interval pattern $\langle [3,4], [7,12], [91,102] \rangle$.

Proof. Consider the subset G of objects covered by the interval pattern $\langle [a_m, b_m] \rangle_{\forall_{m\in\mathcal{M}}}$. According to the closure definition, the lower bound can be expressed as $\forall m \in \mathcal{M}, a_m = \min(\{v_{g,m} \mid g \in G\})$. This is equivalent to $\forall m \in \mathcal{M}, a_m = \min(\{v_{g,m} \mid \forall g \in \mathcal{G}, y_g = 1\})$. Consequently, we deduce that $\forall m \in \mathcal{M}, a_m = \min(\{v \mid \forall g \in \mathcal{G}, (y_g = 1 \implies v = v_{g,m}) \wedge (y_g = 0 \implies v = \mathcal{N}_m^{\uparrow} + 1)\})$. The proof of the upper bound is similar to lower bound.

	y_g	$\underline{H}_{g,1}$	$\underline{H}_{g,2}$	$\underline{H}_{g,3}$
g_1	0	7	13	131
g_2	1	4	12	102
g_3	1	3	7	91
g_4	0	7	13	131
g_5	0	7	13	131
min:		3	7	91

	y_g	$\overline{H}_{g,1}$	$\overline{H}_{g,2}$	$\overline{H}_{g,3}$
g_1	0	1	6	90
g_2	1	4	12	102
g_3	1	3	7	91
g_4	0	1	6	90
g_5	0	1	6	90
max:		4	12	102

Table 2. Values of closure variables $\underline{H}_{g,m}$ and $\overline{H}_{g,m}$ for the running example.

CP4CIP model. FCIP mining can be modeled by the conjunction of the coverage constraints (cf. Eq. 1 and Eq. 2) and the closure constraints (Eqs. 3 to 8). This model uses $2.|\mathcal{M}|$ variables for interval representation, $|\mathcal{G}|$ variables for objects coverage, and $3.|\mathcal{G}|.|\mathcal{M}|$ variables. It involves $|\mathcal{G}|.|\mathcal{M}|$ inclusion constraints, $|\mathcal{G}|$ coverage constraints, and $4.|\mathcal{G}|.|\mathcal{M}|+2.|\mathcal{M}|$ closure constraints. The main limitation of such a model lies in its number of variables and constraints leading to scaling challenges on large datasets.

5 Second Model using a Global Constraint

Similarly to Section 4.1, this second model uses the sets of variables $\underline{X}$ and $\overline{X}$ to represent the lower bounds and the upper bounds of an interval pattern. The $GC4CIP_{\mathcal{N},\theta}(\underline{X}, \overline{X})$ global constraint holds if and only if $\mathcal{V}$ is closed, i.e. $close(\mathcal{V})$ and $\mathcal{V}$ is frequent, i.e. $freq(\mathcal{V})) \geq \theta$. In the following, we describe the new specific filtering rules associated to closure and frequency of an interval pattern.

5.1 Closure filtering rules

Let $\mathcal{V}^* = \langle[\min(\mathcal{D}(\underline{x}_1)), \max(\mathcal{D}(\overline{x}_1))], \ldots, [\min(\mathcal{D}(\underline{x}_{|\mathcal{M}|})), \max(\mathcal{D}(\overline{x}_{|\mathcal{M}|}))]\rangle$ be the largest interval pattern from the domains. Proposition 1 states that values occurring only in objects non covered by $\mathcal{V}^*$ must be removed.

Proposition 1. *Let* $\mathcal{V}^* = \langle[\min(\mathcal{D}(\underline{x}_1)), \max(\mathcal{D}(\overline{x}_1))], \ldots, [\min(\mathcal{D}(\underline{x}_{|\mathcal{M}|})), \max(\mathcal{D}(\overline{x}_{|\mathcal{M}|}))]\rangle$. *Let* $m \in \mathcal{M}$, $g \in \mathcal{G}$,

$$\begin{cases} v_{g,m} \notin \mathcal{D}(\underline{x}_m), \\ v_{g,m} \notin \mathcal{D}(\overline{x}_m) \end{cases} \; if: \; \begin{cases} \exists\, m' \in \mathcal{M}, m \neq m', v_{g,m'} < \min(\mathcal{D}(\underline{x}_{m'})) \;\vee\; v_{g,m'} > \max(\mathcal{D}(\overline{x}_{m'})) \\ \wedge \\ \forall g' \in \mathcal{G},\; g \neq g' \text{ such that } g' \text{ is covered by } \mathcal{V}^*,\; v_{g,m} \neq v_{g',m} \end{cases} \tag{9}$$

Proof. Let m, $m' \in \mathcal{M}$ and $m \neq m'$. Suppose that $v_{g,m} \in \mathcal{D}(\underline{x}_m)$ and $v_{g,m'} < \min(\mathcal{D}(\underline{x}_{m'}))$ or $v_{g,m'} > \max(\mathcal{D}(\overline{x}_{m'}))$. If $v_{g,m'} < \min(\mathcal{D}(\underline{x}_{m'})) \vee v_{g,m'} > \max(\mathcal{D}(\overline{x}_{m'}))$, this means that g is not covered by $\mathcal{V}^*$, i.e. $g \notin cover(\mathcal{V}^*)$. Thus following the closed interval definition in section 2.1, $\underline{x}_m = min(\{v_{g',m} \mid g' \in cover(\mathcal{V}^*)\})$. If there does not exist $g' \in \mathcal{G}$ where g' is covered and $v_{g',m} = v_{g,m}$, then $v_{g,m}$ will never be a bound in $\mathcal{V}^*$, therefore $v_{g,m} \notin \mathcal{D}(\underline{x}_m)$ which contradicts the assumption. The proof for the upper bound is similar.

Example 6. Consider the dataset in Table 1, and the following domains: $\mathcal{D}(\underline{x}_1) = \{4,6\}$, $\mathcal{D}(\overline{x}_1) = \{4,6\}$, $\mathcal{D}(\underline{x}_2) = \{7,8,9,12\}$, $\mathcal{D}(\overline{x}_2) = \{7,8,9,12\}$, $\mathcal{D}(\underline{x}_3) = \{91,101,102,130\}$, $\mathcal{D}(\overline{x}_3) = \{91,101,102,130\}$. Since the values 2 and 3 for the attribute $m1$ are not in $\mathcal{D}(\underline{x}_1)$ and $\mathcal{D}(\overline{x}_1)$, the objects $g1$, $g3$ and $g4$ are not covered. Therefore, the values 7,8,9 will be removed from $\mathcal{D}(\underline{x}_2)$ and $\mathcal{D}(\overline{x}_2)$ because 7 appears in g_3, 8 appears in g_1 and 9 in g_4, and these values do not occur in any covered object.

For defining the next filtering rule, we first consider the joint between the domains of two attributes m and m'. For each value $v_{g,m}$ in $D(x_m)$, we consider the objects g having such a value for attribute m, i.e. $I_m = \{g \mid g \in \mathcal{G}, v_{g,m} \in D(x_m)\}$. Then, for each object g in I_m, we determine its value for the attribute m' So, $join(x_{m'}, x_m) = \{v_{g,m'} \mid v_{g,m'} \in D(x_{m'}), g \in I_m\}$. Then, every value outside the bounds of this set has to be removed.

Proposition 2. *Let $m, m' \in \mathcal{M}$, $m \neq m'$. For simplicity, we denote by x_m as either a lower bound $\underline{x}_m$ or an upper bound $\overline{x}_m$.*

$$\begin{cases} v_{g,m} \notin \mathcal{D}(\underline{x}_m) \text{ if:} & v_{g,m} > max(join(x_{m'}, \underline{x}_m)) \\ v_{g,m} \notin \mathcal{D}(\overline{x}_m) \text{ if:} & v_{g,m} < min(join(x_{m'}, \overline{x}_m)) \end{cases} \tag{10}$$

Proof. Let $m, m' \in \mathcal{M}$ and $m \neq m'$. Suppose that $v_{g,m} \in \mathcal{D}(\underline{x}_m)$ and $v_{g,m} > \max(join(\underline{x}_{m'}, \overline{x}_m))$. $\mathcal{D}(\underline{x}_{m'})$ contains all possible lower bounds for the attribute m', therefore $join(\underline{x}_{m'}, \overline{x}_m)$ returns all the values which are potential lower bounds for the attribute m. If $v_{g,m} > \max(join(\underline{x}_{m'}, \overline{x}_m))$, there exists an object g which is either uncovered or is strictly inside the closed intervals. In both cases, all the values occurring in object g are not located in the closure's lower border. Consequently, the $v_{g,m} \notin \mathcal{D}(\underline{x}_m)$ which contradicts the assumption. The proof for the upper bound is similar.

Example 7. Consider the database in Table 1 and the following domains: $\mathcal{D}(\underline{x}_1) = \{2,3,4,6\}, \mathcal{D}(\overline{x}_1) = \{4,6\}, \mathcal{D}(\underline{x}_2) = \{7,8,9,12\}, \mathcal{D}(\overline{x}_2) = \{7,8,9,12\}, \mathcal{D}(\underline{x}_3) = \{91,101,102,110,130\}$, $\mathcal{D}(\overline{x}_3) = \{91,101,102,110\}$. The objects associated to $\mathcal{D}(\underline{x}_1)$ are g_4 and g_6. The values of these objects for attribute m_3 are 102 and 110. So $join(\overline{x}_1, \underline{x}_3) = \{102,110\}$. In the same way, $join(\overline{x}_1, \overline{x}_3) = \{102,110\}$. Following the filtering rules in 10, value 130 will be removed from $\mathcal{D}(\underline{x}_3)$, because $130 > max(join(\overline{x}_1, \underline{x}_3))$ and values 91, 101 will be removed from $\mathcal{D}(\overline{x}_3)$ since 91 and 101 are smaller than $min(join(\overline{x}_1, \overline{x}_3))$.

5.2 Frequency filtering rules

Values not occurring in any frequent interval pattern have to be removed.

Proposition 3. *Let $m \in \mathcal{M}$, and $\mathcal{V}^p = \langle [\min(\mathcal{D}(\underline{x}_i)), \max(\mathcal{D}(\overline{x}_i))] \rangle_{1 \leq i \neq m \leq |\mathcal{M}|}$ be a partial interval pattern.*

$$\begin{cases} a_m \notin \mathcal{D}(\underline{x}_m) \text{ if:} & freq(\mathcal{V}^p \ +\!\!+ \ [a_m, \max(\mathcal{D}(\overline{x}_m))]) < \theta \\ b_m \notin \mathcal{D}(\overline{x}_m) \text{ if:} & freq(\mathcal{V}^p \ +\!\!+ \ [\min(\mathcal{D}(\underline{x}_m)), b_m]) < \theta \end{cases} \tag{11}$$

Proof. We prove that $a_m \notin \mathcal{D}(\underline{x}_m)$ if the candidate interval on the attribute m using the maximum value as an upper bound will always lead to a frequency less than θ. Suppose that $a_m \in \mathcal{D}(\underline{x}_m)$. We know that $cover(\mathcal{V}^p) \cap cover([a_m, b_m]) = cover(\mathcal{V}^p +\!\!+ [a_m, b_m])$. Since $\mathcal{V}^p$ is a consistent partial solution, $freq(\mathcal{V}^p) \geq \theta$. This means that the frequency of the candidate interval pattern $(\mathcal{V}^p +\!\!+ [a_m, b_m])$ is determined only by $freq([a_m, b_m])$. Consequently, if $freq([a_m, \max(\mathcal{D}(\underline{x}_m))]) < \theta$, then the value a_m as a lower bound will always lead to infrequent candidate interval pattern with current domains. Thus $a_m \notin \mathcal{D}(\underline{x}_m)$ which contradicts the assumption. The proof establishing the inconsistency of the value b_m in the upper bound domain is similar.

Example 8. Consider the dataset in Table 1, a minimum frequency threshold $\theta = 2$, and the following variable domains: $\mathcal{D}(\underline{x}_1) = \{2\}$, $\mathcal{D}(\overline{x}_1) = \{3\}$, $\mathcal{D}(\underline{x}_2) = \{8, 9\}$, $\mathcal{D}(\overline{x}_2) = \{8, 9, 12\}$, $\mathcal{D}(\underline{x}_3) = \{91, 101, 102, 130\}$, $\mathcal{D}(\overline{x}_3) = \{91, 101, 102, 130\}$. Following the frequency filtering rule, and considering the partial assignement $\mathcal{V}^p = \langle [2,\ 3],\ [91,\ 130] \rangle$, the values 8 and 9 will be removed from $\mathcal{D}(\underline{x}_2)$ and $\mathcal{D}(\overline{x}_2)$ respectively, since $freq(\mathcal{V}^p +\!\!+ [9, max(\mathcal{D}(\overline{x}_2))]) < 2$ and $freq(\mathcal{V}^p +\!\!+ [min(\mathcal{D}(\underline{x}_2)), 8]) < 2$ (Equation 11)

GC4CIP model. Our global constraint $\text{GC4CIP}_{\mathcal{N},\theta}(\underline{X}, \overline{X})$ can be used with additional constraints to handle more complex mining tasks (see Section 7.3).

6 GC4CIP Filtering Algorithm

In this section, we introduce our algorithm designed to implement the filtering rules outlined in Section 5, ensuring the domain consistency of the GC4CIP constraint. Our filtering algorithm leverages an internal data structure to certify candidate solutions and optimize the filtering process.

A specific data structure. Consider a tree represented as $T = (N,\ E,\ r,\ inf,\ sup)$ where N is the set of nodes in T. Each node $n \in N$ contains an interval $[a_n, b_n]$ with a_n and b_n being values returned by the operators $inf(n)$ and $sup(n)$ respectively. The root node of T is r, and E is the set of edges in the tree. The collection of such trees is called forest and denoted as F. In our algorithm, this forest is represented as a list of trees, where each tree is associated with an attribute $m \in \mathcal{M}$ from the dataset. Additionally, we introduce a Boolean array, denoted as Cov, designed to compute the current coverage of the set of objects (i.e. leaf nodes). For $g \in \mathcal{G}$, $Cov[g]$ is set to 1 if g can be covered, otherwise it is set to 0. The synchronization of trees within the forest is maintained through the coverage array.

Initially, for each attribute m and object g in the dataset, a leaf node $n \in N$ is created, such that $n = [v_{g,m}, v_{g,m}]$ where $v_{g,m} \in \mathcal{N}_m$. A parent node p is created over a subset of leaf nodes, where $inf(p)$ is set to the minimum a_n of its children and $sup(p)$ is set to the maximum b_n of its children. The interval $[a_r, b_r]$ associated with the root node of each tree establishes coherent bounds for the variables corresponding to the attribute $m \in \mathcal{M}$ (i.e. $\underline{x}_m$ and $\overline{x}_m$).

Algorithm 1: GC4CIP coverage update

```
1  Function PushDown(T: tree, depth, n: current node, a: value in N_m, b: value in N_m,
   Cov: coverage array)
2      if depth = maxdepth(T) ∧ (inf(n) < a ∨ sup(n) > b) then
3          inf(n) ⟵ +∞;   sup(n) ⟵ −∞;
4          Cov[ObjectIndexFromLeaf(n)] ⟵ 0;
5      else
6          foreach c ∈ child(n) do
7              if inf(c) < a ∧ sup(c) > b then
8                  PushDown(T, c, a, b, depth+1)
9              else if inf(c) < a then
10                 PushDown(T, c, a, sup(c), depth+1)
11             else if sup(c) > b then
12                 PushDown(T, c, inf(c), b, depth+1)
```

To maintain the trees and the coverage array with respect to changes in the variables domains, we introduce a function *PushDown*(), see Algorithm 1. The *PushDown* function allows to update leaves nodes having their intervals (i.e. $[v_{g,m}, v_{g,m}]$) not included in the interval of the root node, lines 3 and 12. If this condition holds, the interval of the leaf node is set to $[+\infty, -\infty]$, making the corresponding object uncovered.

Implementing the filtering rules. Algorithm 2 applies the filtering rules described in Section 5. When the domain of a bound variable of an interval x_m (i.e. $\underline{x}_m$ or $\overline{x}_m$) changes, we first update the leaves through the *PushDown*() function described in Algorithm 1. The leaves of all the trees within the forest are then synchronized through the cover array such that each leaf corresponding to a newly uncovered object is set to $[+\infty, -\infty]$, lines 6 to 8 in Algorithm 2. The new interval borders are updated using a bottom-up approach which sets for each parent node a new lower and upper bound consisting respectively, of the smallest value from its children's lower bound and the highest value among its children's upper bound, lines 10 to 13. The intervals coherence is maintained in lines 14 and 15 by filtering values that are greater than $sup(r)$ and smaller than $inf(r)$ from both $\mathcal{D}(\underline{x}_{m'})$ and $\mathcal{D}(\overline{x}_{m'})$. Lines 16 and 17 implements Proposition 1 using the cover array by filtering values that appears only in uncovered objects. Afterward lines 19 and 20 are a straightforward application of Proposition 2 as they filter all the values greater than $max(join(x_m, \underline{x}_{m'}))$ and smaller than $min(join(x_m, \overline{x}_{m'}))$. Finally, Lines 22 and 23 implements Proposition 3 since it filters values that only leads to infrequent closed interval patterns.

Time complexity analysis Algorithm 1 has a worst-case time complexity of $O(|\mathcal{G}|)$, which correspond to the number of leaves in the tree. This is simplified from $O(S^{log_S|\mathcal{G}|})$, where S is the maximal number of children of a parent node. The time complexity of the algorithm 2 in the worst case is given by $O(|\mathcal{G}| + (|\mathcal{M}| \cdot |\mathcal{G}|^3 \cdot log_S|\mathcal{G}|) + |\mathcal{G}|)$. Consequently the complexity of GC4CIP is bounded by $O(|\mathcal{M}| \cdot |\mathcal{G}|^3 \cdot log_S|\mathcal{G}|)$.

Algorithm 2: GC4CIP closure filtering

Input: $\underline{X}_{\mathcal{M}} = \{\underline{x}_1, \underline{x}_2, ..., \underline{x}_{|\mathcal{M}|}\}$, $\overline{X}_{\mathcal{M}} = \{\overline{x}_1, \overline{x}_2, ..., \overline{x}_{|\mathcal{M}|}\}$, θ :Frequency threshold, $x_{m'}$: modified variable, $F = [T_1, ..., T_{|M|}]$: Forest corresponding to databases' attributes, $Cov[|\mathcal{G}|]$:Coverage vector
Output: Consistent $\underline{X}_{\mathcal{M}} = \{\underline{x}_1, \underline{x}_2, ..., \underline{x}_{|\mathcal{M}|}\}$ and $\overline{X}_{\mathcal{M}} = \{\overline{x}_1, \overline{x}_2, ..., \overline{x}_{|\mathcal{M}|}\}$
begin:
PushDown(F[m'],depth=0, $r \in$ F[m], $min(\underline{x}_{m'})$, $max(\overline{x}_{m'})$)
foreach $m \in \mathcal{M}$ **do**
  `// synchronize the tree leaves`
  foreach *leaf* $n \in F[m]$ **do**
    if $Cov[n] = 0$ **then**
      $inf(n) \longleftarrow +\infty$; $sup(n) \longleftarrow -\infty$
  `// update nodes intervals of the tree`
  $level \longleftarrow Maxdepth(F[m]) - 1$
  while $level \geq 0$ **do**
    foreach $n \in F[m]$ *where* $depth(n) = level$ **do**
      $inf(n) \longleftarrow min(\{inf(c) \mid c \in child(n)\})$
      $sup(n) \longleftarrow max(\{sup(c) \mid c \in child(n)\})$
    $level \longleftarrow level - 1$
  `// Maintaining interval coherence`
  $\mathcal{D}(x_m) \longleftarrow \mathcal{D}(x_m) \setminus \{v \mid v \in \mathcal{D}(x_m) \wedge v > sup(r)\}$
  $\mathcal{D}(x_m) \longleftarrow \mathcal{D}(x_m) \setminus \{v \mid v \in \mathcal{D}(x_m) \wedge v < inf(r)\}$
  `// Closure filtering`
  $\mathcal{D}(\overline{x}_m) \longleftarrow \mathcal{D}(\overline{x}_m) \setminus \{v \mid v \in \mathcal{D}(\overline{x}_m) \wedge \forall g \in \mathcal{G}, v \neq v_{g,m}$ where $Cov[g] = 1\}$
  $\mathcal{D}(\underline{x}_m) \longleftarrow \mathcal{D}(\underline{x}_m) \setminus \{v \mid v \in \mathcal{D}(\underline{x}_m) \wedge \forall g \in \mathcal{G}, v \neq v_{g,m}$ where $Cov[g] = 1\}$
  if $m \neq m'$ **then**
    $\mathcal{D}(\overline{x}_m) \longleftarrow \mathcal{D}(\overline{x}_m) \setminus \{v \mid v \in \mathcal{D}(\overline{x}_m) \wedge v < min(join(x_{m'}, \overline{x}_m))\}$
    $\mathcal{D}(\underline{x}_m) \longleftarrow \mathcal{D}(\underline{x}_m) \setminus \{v \mid v \in \mathcal{D}(\underline{x}_m) \wedge v > max(join(x_{m'}, \underline{x}_m))\}$
`// Frequency filtering`
$\mathcal{V}^p \longleftarrow \langle [\min(\mathcal{D}(\underline{x}_i)), \max(\mathcal{D}(\overline{x}_i))] \rangle_{1 \leq i \neq m' \leq |\mathcal{M}|}$
$\mathcal{D}(\underline{x}_{m'}) \longleftarrow \mathcal{D}(\underline{x}_{m'}) \setminus \{v \mid v \in \mathcal{D}(\underline{x}_{m'}) \wedge freq(\mathcal{V}^p +\!\!+ [v, max(\overline{x}_{m'})]) < \theta\}$
$\mathcal{D}(\overline{x}_{m'}) \longleftarrow \mathcal{D}(\overline{x}_{m'}) \setminus \{v \mid v \in \mathcal{D}(\overline{x}_{m'}) \wedge freq(\mathcal{V}^p +\!\!+ [min(\underline{x}_{m'}), v]) < \theta\}$

7 Experiments and results

The experimental evaluation is designed to address the following questions:
1. What are the results in terms of CPU time for our two models compared to :

(a) CP approaches such as CP4IM and CLOSEDPATTERN outlined in section 3 which require a pre and post processing step ?

(b) the ad hoc method MINTCHANGE described in Section 3?

2. Can our CP model be seamlessly extended to address other data mining tasks on numerical data (e.g. clustering) while being competitive with ad hoc approaches (e.g. K-MEANS) ?

	NT	AP	BK	Cancer	CH	Yacht	LW
$\|\mathcal{M}\|$	3	5	5	9	8	7	10
$\|\mathcal{G}\|$	130	135	96	116	209	308	189
distinct values	67	674	313	900	396	322	253
	Interordinal scaled datasets						
|Binary attributes|	134	1348	626	1800	792	644	506
density (%)	52.23	50.37	50.79	50.50	51.01	51.08	51.97

Table 3. Datasets characteristics

7.1 Benchmark datasets

We selected several difficult numerical datasets for declarative approaches which were used in [8]. We also selected two other datasets (Cancer and Yacht) from the UC Irvine archive [1]. The names of datasets are given by standard abbreviations used in the database of Bilkent University. All the selected datasets come in different sizes and types, most of them containing real values and one of them negative values. Table 3 provides a comprehensive summary of dataset characteristics, including the following key metrics: the number of objects denoted by $|\mathcal{G}|$, the number of items represented by $|\mathcal{M}|$, the sum of distinct values which is calculated by the formula $\sum_{m=1}^{|\mathcal{M}|} |\mathcal{N}_m|$, the count of binary attributes resulting from the process of interordinal scaling's binarization denoted as |Binary attributes| and the density of the binarized datasets. The dataset's density, after applying IS method, is always greater than 50 %. This is a direct result of the IS method which assigns for each numerical value at least one binary attribute to 1.

The implementation of our approach was carried out using the `OR-tools` solver v9.0[2]. All experiments were conducted on an AMD Opteron 6174 with 2,2 GHz of CPU and 256 GB of RAM. with a timeout of 12 hours. For each dataset, we decreased the (relative) frequency threshold until it is impossible to extract all closed interval patterns within the allocated time/memory. The source code and datasets are available at `https://github.com/djawed-bkh/CPAIOR2024`.

7.2 Mining CIP

Comparing with other CP approaches. Table 4 presents the computation time required to discover all closed interval patterns under different minimum frequency thresholds across various datasets. In these experiments, for both the CP4IM and CLOSEDPATTERN approaches, we provide the cpu-time of pre-processing and post-processing steps for using itemset mining. The pre-processing time is negligible compared to the post-processing time due to the high number of closed itemsets found.

If we consider the two reified models, CP4CIP and CP4IM, we can observe that CP4CIP consistently outperforms CP4IM across most of the selected datasets. In terms of CPU times, an average speed-up of 8.36, 10.11, 14.12, and 32.64 is observed for the BK, CH, Cancer, and AP datasets, respectively. For the LW, and Yacht datasets, we

[1] https://archive.ics.uci.edu/datasets

[2] https://github.com/google/or-tools/

$\mathcal{N}$	θ	# Sol	Time (s)						
	(%)	(≈)	**(1)**	**(2)**	**(3)**	**(1 + 3)**	**(2 + 3)**	**(4)**	**(5)**
BK	80	10^6	1840.21	148.91	176.65	2016.86	325.56	271.10	89.63
	70	10^7	15132.87	1457.99	1326.58	16459.45	2784.57	1770.22	655.63
	60	10^7	TO	8643.34	6713.25	TO	15356.59	7311.24	2879.54
	50	10^8	TO	28302.62	19307.70	TO	47610.32	18471.23	7780.65
	20	10^8	TO	TO	TO	TO	TO	TO	34598.10
Cancer	95	10^4	170.14	6.19	13.69	183.83	19.88	18.42	5.80
	94	10^5	568.00	18.21	38.88	606.88	57.09	45.43	15.66
	92	10^5	6944.07	294.14	542.82	7486.89	836.96	486.87	190.84
	90	10^6	29787.19	1190.42	2348.45	32135.64	3538.87	1806.19	786.25
AP	80	10^5	783.92	175.02	55.21	839.13	230.23	28.55	19.18
	70	10^6	5909.86	189.30	415.76	6325.62	605.06	194.64	128.83
	60	10^6	18479.87	7995.84	1275.85	19755.72	9271.69	548.12	373.01
	50	10^7	TO	23252.89	2964.71	TO	26217.60	1223.79	770.83
	20	10^7	TO	43199.73	3052.93	TO	46252.66	5129.20	2891.55
	0	10^7	TO	TO	TO	TO	TO	5867.37	2343.98
CH	95	10^6	25.59	1.16	29.93	55.52	31.09	5.98	1.60
	90	10^5	608.94	36.58	224.70	833.64	261.28	89.81	38.42
	85	10^6	4753.35	331.08	835.24	5588.59	1166.32	671.49	256.86
	80	10^6	19154.96	1444.64	18009.40	37164.36	19454.04	2739.85	890.82
	50	TO	TO	TO	TO	TO	TO	TO	TO
LW	80	10^6	1612.68	96.91	174.46	1787.14	271.37	1638.03	181.81
	70	10^6	12904.12	757.02	1279.34	14183.34	2036.36	9886.90	1269.50
	60	10^7	TO	3436.91	5236.91	TO	8673.82	33 148.24	4,965.20
	50	10^8	TO	11060.23	15588.10	TO	26648.33	TO	14298.64
	20	TO	TO	TO	TO	TO	TO	TO	TO
NT	80	10^3	0.87	0.06	0.07	0.97	0.13	1.80	0.13
	50	10^4	7.08	0.41	0.50	7.58	0.91	11.01	0.91
	20	10^4	28.13	1.53	1.83	29.96	3.36	28.77	2.89
	10	10^5	41.75	2.51	2.61	44.36	5.12	32.50	4.02
	0	10^5	62.48	2.88	3.13	65.61	6.01	33.72	3.81
Yacht	80	10^4	40.12	2.03	83.20	123.32	85.23	90.92	2.45
	50	10^6	7277.85	336.03	268.28	7546.13	604.31	4090.63	181.63
	40	10^6	30519.66	1282.32	727.09	31246.75	2009.41	9380.16	501.52
	30	10^7	TO	4265.71	1695.63	TO	5961.34	20464.22	1179.13
	20	10^7	TO	12898.20	2874.08	TO	15772.28	33294.36	2487.68
	0	10^7	TO	TO	TO	TO	TO	TO	4116.60

(1): CP4IM **(2):** CLOSEDPATTERN
(3): Preprocessing + Postprocessing **(4):** CP4CIP **(5):** GC4CIP

Table 4. Closed Interval Pattern Mining Methods vs Itemset Mining Methods with Interordinal Scaling

observe a speedup of 1.26 and 2.17 respectively. Finally in the NT dataset the results are more balanced as in the high frequencies CP4IM slightly overcome CP4CIP whereas in low frequencies CP4CIP is better. This can be explained with the low number of distinct values, resulting in fewer binary attributes generated with the IS method which enables CP4IM to enhance performance.

If we consider the use of global constraints GC4CIP and CLOSEDPATTERN, GC4CIP consistently outperforms CLOSEDPATTERN across all datasets. Regarding CPU times, an average speed-up of 1.20, 1.67, 3.98, and 4.83 is noted for the NT, LW, Cancer, and BK datasets, respectively. For the Yacht, CH, and AP datasets, a substantial speed-up of 10.70, 13.15, and 18.31 is observed, showcasing the efficiency of GC4CIP which emerges as the most efficient approach among all the compared declarative methods. Its

main competitor, CLOSEDPATTERN, is consistently outperformed in the majority of instances, even when comparing only resolution times (without including pre-processing and post-processing times).

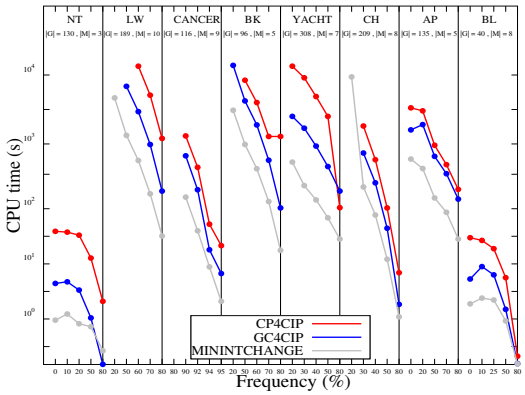

Fig. 1. MinIntChange compared to declarative approaches

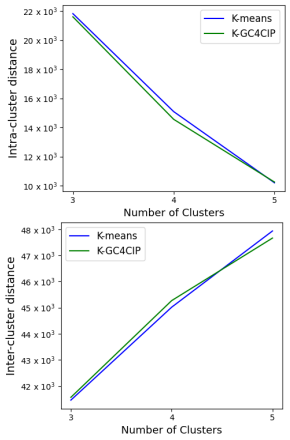

Fig. 2. Cluster quality of GC4CIP and K-MEANS

Comparing with the ad hoc approach MININTCHANGE. Figure 1 compares CPU times of our two proposals with the adhoc method MININTCHANGE. As expected, MININTCHANGE outperforms the declarative approaches across all databases, with speedup between GC4CIP and MININTCHANGE ranging from 2 for the NT database to 7 in Yacht. This outcome is due to the dedicated character of MININTCHANGE, tailored specifically for the FCIP mining task. However, in contrast to our generic declarative approaches, MININTCHANGE struggles to adapt to other data mining tasks due to its need of rewriting the solving algorithm. In the following section, we illustrate the genericity of our declarative approaches by applying them to another data mining task.

7.3 Modelling a k-clustering problem using GC4CIP

We demonstrate the genericity of our approach by considering a clustering task as a use case. Our objective is to find a clustering which forms a partition over the objects in the dataset as akin to K-MEANS method. Closed interval patterns form the set of clusters, making this task known as conceptual clustering, where each cluster is defined by a unique concept (i.e. closure property). This task of finding k interval closed patterns $\{\mathcal{V}^1, ..., \mathcal{V}^k\}$ is formally described by the following:

$$\begin{cases} \textbf{Closure} & close(\mathcal{V}^i) & , \forall 1 \leq i \leq k \\ \textbf{No overlapping} & cover(\mathcal{V}^i) \cap cover(\mathcal{V}^j) = \varnothing & , \forall 1 \leq i < j \leq k \\ \textbf{Total coverage} & \bigcup_{1 \leq i \leq k} cover(\mathcal{V}^i) = \mathcal{G} \end{cases}$$

To carry this task, we extend our global constraint GC4CIP to handle the conceptual clustering directly from the numerical data. This involves adding a new set of variables denoted Y as parameter in our global constraint i.e $GC4CIP_{\mathcal{N},\theta}(\underline{X}, \overline{X}, Y)$. The set of binary variables $Y = \{y_1, ..., y_{|\mathcal{G}|}\}$ indicates whether an object is covered by $\mathcal{V}$ or not. The filtering rules over the variables Y in the global constraint are described in supplementary material `https://github.com/djawed-bkh/CPAIOR2024`. Our CP based k-Clustering model requires three sets of variables. $\underline{X}^i$ and $\overline{X}^i$ to represent each interval pattern associated to a cluster and Y^i to indicate whether an object is included in a cluster. In term of constraints, our model requires the conjunction of three distinct types of constraints. The first one is a closure constraint, represented by $\text{GC4CIP}_{\mathcal{N},\theta}(\underline{X}^i, \overline{X}^i, Y^i)$ which is applied k times to generate k closed interval patterns each one representing a cluster. Then, to ensure the non-overlapping coverage of our interval patterns, we introduce a partitioning constraint which ensures that each object $g \in \mathcal{G}$ can be covered with at most one interval pattern, thus guaranteeing that an object belongs to a single clustering at most (i.e. $\sum_{1 \leq i \leq k} y_g^i = 1$). Finally, we introduce a constraint enforcing the k-clusters to cover all the objects in the database (i.e. $\bigcup_{1 \leq i \leq k} cover(\mathcal{V}^i) = \mathcal{G}$).

In Figure 2, we showcase the results of clustering on the NT database with 3, 4, and 5 clusters for both the K-MEANS and K-GC4CIP approaches, with a 12 hours timeout. To measure the results quality we use the intra-cluster and inter-cluster Euclidean distances. Notably, for cluster numbers 3 and 4, the K-GC4CIP method outperforms K-MEANS in both intra-cluster distance and inter-cluster distance as the former is smaller and the latter is greater than the heuristic approach. However, for a number of clusters of 5, we note that K-MEANS performed slightly better than K-GC4CIP in both measures, which can be explained by the timeout of K-GC4CIP, as it was unable achieve an exhaustive search within the allocated time.

8 Conclusion

In this paper, we introduced two CP models (CP4CIP and GC4CIP) for mining FCIP directly from numerical data without requiring any pre- or post-processing step. We demonstrated the efficiency of GC4CIP compared to existing declarative methods and an ad hoc one. Finally, we illustrated the genericity of our approach by combining GC4CIP with other constraints to tackle the conceptual k-clustering problem. The declarative nature of our contributions enables straightforward combination with other declarative approaches. For example, combining GC4CIP with CP4IM allows effortless mining of closed patterns from heterogeneous data, where the former handles numerical data and the later focuses on binary data.

Acknowledgement. The first author is supported by the French National Research Agency (ANR) and Region Normandie under grant HAISCoDe.

References

1. Belfodil, A., Kuznetsov, S.O., Robardet, C., Kaytoue, M.: Mining convex polygon patterns with formal concept analysis. In: Sierra, C. (ed.) IJCAI. pp. 1425–1432 (2017)
2. Calders, T., Rigotti, C., Boulicaut, J.F.: A survey on condensed representations for frequent sets. In: Constraint-Based Mining and Inductive Databases. Lecture Notes in Computer Science, vol. 3848, pp. 64–80. Springer (2005)
3. Chabert, M., Solnon, C.: A global constraint for the exact cover problem: Application to conceptual clustering. J. Artif. Intell. Res. **67**, 509–547 (2020)
4. Codocedo, V., Napoli, A.: A Proposition for Combining Pattern Structures and Relational Concept Analysis. In: ICFCA. pp. 96 – 111 (2014)
5. Dao, T., Vrain, C., Duong, K., Davidson, I.: A framework for actionable clustering using constraint programming. In: ECAI. pp. 453–461. Frontiers in Artificial Intelligence and Applications (2016)
6. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Machine learning: Proceeding of the twelfth international conference. pp. 194–202. Morgan Kaufmann (1995)
7. Guns, T., Nijssen, S., De Raedt, L.: k-pattern set mining under constraints. IEEE Transactions on Knowledge and Data Engineering **25**(2), 402–418 (2013)
8. Kaytoue, M., Kuznetsov, S., Napoli, A.: Revisiting numerical pattern mining with formal concept analysis. IJCAI (11 2011)
9. Khiari, M., Boizumault, P., Crémilleux, B.: Constraint programming for mining n-ary patterns. In: CP. pp. 552–567. Springer (2010)
10. Lazaar, N., Lebbah, Y., Loudni, S., Maamar, M., Lemière, V., Bessiere, C., Boizumault, P.: A global constraint for closed frequent pattern mining. In: Rueher, M. (ed.) CP. pp. 333–349. Springer (2016)
11. Makhalova, T., Kuznetsov, S.O., Napoli, A.: Mint: Mdl-based approach for mining interesting numerical pattern sets. Data Min. Knowl. Discov. pp. 108–145 (2022)
12. Meeng, M., Knobbe, A.J.: For real: a thorough look at numeric attributes in subgroup discovery. Data Min. Knowl. Discov. **35**(1), 158–212 (2021)
13. Millot, A., Cazabet, R., Boulicaut, J.: Optimal subgroup discovery in purely numerical data. In: PAKDD. Lecture Notes in Computer Science, vol. 12085, pp. 112–124. Springer (2020). https://doi.org/10.1007/978-3-030-47436-2_9
14. Nguyen, H.V., Vreeken, J.: Flexibly mining better subgroups. In: Venkatasubramanian, S.C., Jr., W.M. (eds.) Proceedings of the SIAM International Conference on Data Mining, USA,. pp. 585–593. SIAM (2016). https://doi.org/10.1137/1.9781611974348.66
15. Nijssen, S., Zimmermann, A.: Constraint-based pattern mining. In: Frequent Pattern Mining, pp. 147–163. Springer (2014). https://doi.org/10.1007/978-3-319-07821-2_7
16. Raedt, L.D., Guns, T., Nijssen, S.: Constraint programming for data mining and machine learning. In: AAAI (2010)
17. Salleb-Aouissi, A., Vrain, C., Nortet, C.: Quantminer: A genetic algorithm for mining quantitative association rules. In: Veloso, M.M. (ed.) IJCAI. pp. 1035–1040 (2007)
18. Song, C., Ge, T.: Discovering and managing quantitative association rules. In: CIKM'13. pp. 2429–2434 (2013)
19. Uno, T., Asai, T., Uchida, Y., Arimura, H.: LCM: an efficient algorithm for enumerating frequent closed item sets. In: Proceedings of the ICDM Workshop on Frequent Itemset Mining Implementations, (2003)
20. Witteveen, J., Duivesteijn, W., Knobbe, A.J., Grünwald, P.: Realkrimp - finding hyperintervals that compress with MDL for real-valued data. In: IDA. pp. 368–379 (2014)