Supplementary Material

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

1 Filtering rules based on coverage variables

In this section we describe the filtering rules used to extend GC4CIP by adding the set of variables Y as parameter (i.e. $\text{GC4CIP}_{\mathcal{N},\theta}(\underline{X}, \overline{X}, Y)$).

1.1 From coverage variables to interval borders variables

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

$$\begin{cases} v_{g,m} \notin \mathcal{D}(\underline{x}_m), \\ v_{g,m} \notin \mathcal{D}(\overline{x}_m) \end{cases} \; if: \begin{cases} \mathcal{D}(y_g) = \{0\} \\ \wedge \\ \nexists g' \text{ where } \mathcal{D}(y_{g'}) = \{1\} \wedge v_{g,m} = v_{g',m} \end{cases} \tag{1}$$

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. $\mathcal{D}(y_g) = \{0\}$. 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 i.e. $\mathcal{D}(y_{g'}) = \{1\}$ 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.

	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 $\mathcal{N}$

Example 1. Consider the dataset in Table 1, the variables domain representing intervals: $\mathcal{D}(\underline{x}_1) = \{4,6\}$, $\mathcal{D}(\overline{x}_1) = \{4,6\}$, $\mathcal{D}(\underline{x}_2) = \{7,8,9,12\}$, $\mathcal{D}(\overline{x}_2) = \{7,8,9,12\}$, $\mathcal{D}(\underline{x}_3) = \{91,101,102,130\}$, $\mathcal{D}(\overline{x}_3) = \{91,101,102,130\}$ and the variables domains representing coverage : $\mathcal{D}(y_{g_1}) = \{0\}$, $\mathcal{D}(y_{g_2}) = \{1\}$, $\mathcal{D}(y_{g_3}) = \{0\}$, $\mathcal{D}(y_{g_4}) = \{0\}$, $\mathcal{D}(y_{g_5}) = \{1\}$. Since the objects g_1, g_3, g_4 are not covered i.e. $\mathcal{D}(y_{g_1}) = \mathcal{D}(y_{g_3}) = \mathcal{D}(y_{g_4}) = \{0\}$, 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.

1.2 From interval borders variables to coverage variables

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 2 states that an object must be uncovered if it contains a value occurring outside the interval borders of $\mathcal{V}^*$.

Proposition 2 Let $\mathcal{V}^* = \langle[min(\mathcal{D}(\underline{x}_1)), max(\mathcal{D}(\overline{x}_1))], ..., [min(\mathcal{D}(\underline{x}_{|\mathcal{M}|})), max(\mathcal{D}(\overline{x}_{|\mathcal{M}|}))]\rangle$. Let $g \in \mathcal{G}$

$$1 \notin \mathcal{D}(y_g) \text{ if } \exists m \in \mathcal{M} \text{ s.t.} \begin{cases} v_{g,m} < min(\mathcal{D}(\underline{x}_m)) \\ \vee \\ v_{g,m} > max(\mathcal{D}(\overline{x}_m)) \end{cases} \tag{2}$$

Proof. Assume that $\mathcal{D}(y_g) = \{1\}$ and there exist a value $v_{g,m}$ such that $v_{g,m} < min(\mathcal{D}(\underline{x}_m))$ or $v_{g,m} > max(\mathcal{D}(\overline{x}_m))$. if $v_{g,m} < min(\mathcal{D}(\underline{x}_m))$ or $v_{g,m} > max(\mathcal{D}(\overline{x}_m))$ this means that the object g is not an occurrence of $\mathcal{V}^*$ i.e. $\mathcal{B}[g] \not\sqsubseteq \mathcal{V}^*$. Following the cover definition in Section 2 of the paper $cover(\mathcal{V}^*) = \{g \mid \mathcal{B}[g] \sqsubseteq \mathcal{V}^*\}$, g is not covered by $\mathcal{V}^*$ as $\mathcal{B}[g]$ is not an occurrence of $\mathcal{V}^*$. Therefore $\mathcal{D}(y_g) = \{0\}$ wich contradict the assumption.

Example 2. Consider the dataset in Table 1, the variables domain representing intervals: $\mathcal{D}(\underline{x}_1) = \{2,3,4\}$, $\mathcal{D}(\overline{x}_1) = \{2,3,4\}$, $\mathcal{D}(\underline{x}_2) = \{7,8,9,12\}$, $\mathcal{D}(\overline{x}_2) = \{7,8,9,12\}$, $\mathcal{D}(\underline{x}_3) = \{91,101,102,130\}$, $\mathcal{D}(\overline{x}_3) = \{91,101,102,130\}$ and the variables domains representing coverage : $\mathcal{D}(y_{g_1}) = \{1\}$, $\mathcal{D}(y_{g_2}) = \{1\}$, $\mathcal{D}(y_{g_3}) = \{1\}$, $\mathcal{D}(y_{g_4}) = \{1\}$, $\mathcal{D}(y_{g_5}) = \{0,1\}$. Since the value 6 is greater than $max(\mathcal{D}(\overline{x}_m))$, the value 1 will be removed from $\mathcal{D}(y_{g_5})$.

1.3 Between coverage variables

Proposition 3 states that an object is covered if all the values occurring in it are included between values of two other covered objects.

Proposition 3 Let $\mathcal{V}^* = \langle[min(\mathcal{D}(\underline{x}_1)), max(\mathcal{D}(\overline{x}_1))], ..., [min(\mathcal{D}(\underline{x}_{|\mathcal{M}|})), max(\mathcal{D}(\overline{x}_{|\mathcal{M}|}))]\rangle$ and let $g, g', g'' \in \mathcal{G}$.

$$0 \notin \mathcal{D}(y_g) \text{ if } \forall m \in \mathcal{M} \ \exists\ g \text{ s.t.} \begin{cases} v_{g,m} \geq min(\{v_{g',m} \mid g' \neq g,\ \mathcal{D}(y_{g'}) = \{1\}\}) \\ \wedge \\ v_{g,m} \leq max(\{v_{g'',m} \mid g'' \neq g,\ \mathcal{D}(y_{g''}) = \{1\}\}) \end{cases} \tag{3}$$

Proof. Let $g, g', g'' \in \mathcal{G}$ such that $g \neq g' \neq g''$ where g', g'' are covered i.e. $\mathcal{D}(y_{g'}) = \mathcal{D}(y_{g''}) = \{1\}$ and g is not instantiated yet i.e. $\mathcal{D}(y_g) = \{0,1\}$.
Suppose that $\mathcal{D}(y_g) = \{0\}$ and for all $m \in \mathcal{M}$ $v_{g',m} \leq v_{g,m} \leq v_{g'',m}$. Having g' and g'' covered mean that $\mathcal{B}[g']$ and $\mathcal{B}[g'']$ are occurrences of $\mathcal{V}^*$. Therefore, since $v_{g',m} \leq v_{g,m} \leq v_{g'',m}$ for all $m \in \mathcal{M}$, g is also an occurence of $\mathcal{V}^*$, which means that g is covered (i.e. $\mathcal{D}(y_g) = \{1\}$) which contradict the assumption.

Example 3. Consider the database in Table 1, we suppose that during the search, the domain of the Y variables are as fellows:$\mathcal{D}(y_1) = \{1\}$, $\mathcal{D}(y_2) = \{0,1\}$, $\mathcal{D}(y_3) = \{1\}$, $\mathcal{D}(y_4) = \{0,1\}$, $\mathcal{D}(y_5) = \{1\}$. Following the filtering rule 3, the value 0 will be filtered from $\mathcal{D}(y_2)$ and $\mathcal{D}(y_4)$ since $\forall m \in \mathcal{M}$ $v_{2,m},\ v_{4,m} \in \langle [2,6], [7,12], [91,130] \rangle$.

Proposition 4 states that an object $g \in \mathcal{G}$ is uncovered if there exist an uncovered object $g' \in \mathcal{G}$ having it values included between the values of g and the values of a covered object $g'' \in \mathcal{G}$.

Proposition 4 Let $g,\ g',\ g'' \in \mathcal{G}$ where $g \neq g' \neq g''$.

$$1 \notin \mathcal{D}(y_g) \text{ if } \begin{cases} \mathcal{D}(y_{g'}) = \{0\} \wedge \mathcal{D}(y_{g''}) = \{1\} \\ \wedge \\ \forall m \in \mathcal{M},\ v_{g,m} \leq v_{g',m} < v_{g'',m} \ \vee \ v_{g'',m} < v_{g',m} \leq v_{g,m} \end{cases} \tag{4}$$

Proof. Suppose that $\mathcal{D}(y_g) = \{1\}$ and $v_{g,m} \leq v_{g',m} < v_{g'',m} \ \vee \ v_{g'',m} < v_{g',m} \leq v_{g,m}$ $\forall m \in \mathcal{M}$. $\mathcal{D}(y_g) = \{1\}$ mean that g is covered by $\mathcal{V}^*$. Following the definition of cover in Section 2 of the paper $cover(\mathcal{V}^*) = \{g \mid \mathcal{B}[g] \sqsubseteq \mathcal{V}^*\}$, all the objects that are occurrences of $\mathcal{V}^*$ must be covered which means that there does not exist an uncovered object that is an occurrence of $\mathcal{V}^*$. Having g, g'' being occurrences of $\mathcal{V}^*$ and $v_{g,m} \leq v_{g',m} < v_{g'',m} \ \vee \ v_{g'',m} < v_{g',m} \leq v_{g,m}$ $\forall m \in \mathcal{M}$ mean that g' is also an occurrence of $\mathcal{V}^*$, but $\mathcal{D}(y'_g) = \{0\}$, therefore g can not be covered i.e. $1 \notin \mathcal{D}(y_g)$ (which contradict the assumption) as there exist an uncovered object g' that has values included between g and g'' values'.

Example 4. Consider the dataset in Table 1, we suppose that during the search, the domain of the Y variables are as fellows:$\mathcal{D}(y_1) = \{0\}$, $\mathcal{D}(y_2) = \{0\}$, $\mathcal{D}(y_3) = \{0\}$, $\mathcal{D}(y_4) = \{1\}$, $\mathcal{D}(y_5) = \{0,1\}$. Following the filtering rule 4, the value 1 will be filtered from $\mathcal{D}(y_5)$ because there exists objects g_2 and g_4 such that for all $m \in \mathcal{M},\ v_{5,m} \geq v_{2,m}$ and $v_{2,m} > v_{4,m}$.

2 Tree Search Comparison

In Table 2 , we present a comparative analysis of tree search sizes and failure occurrences across all the databases. Firstly, we compare the two reified models (CP4CIP) and CP4IM. We observe that, for the BK, and AP datasets, CP4CIP generates trees that are respectively 0.55% and 43.55% smaller than those produced by CP4IM, with a significant reduction in failure occurrences. However, in the NT, CH, Yacht and Cancer datasets, we note more balanced results. In these cases, CP4CIP generates smaller tree searches at lower frequencies (0% and 10% threshold for NT, 80% and 85% threshold for CH, and 90% for Cancer), while CP4IM produces smaller tree searches at higher frequencies. In the LW database, CP4IM generate a tree that is 70% smaller than CP4CIP.

$\mathcal{N}$	θ	# Sol	#Nodes				#Failures			
	(%)	(≈)	**(1)**	**(2)**	**(3)**	**(4)**	**(1)**	**(2)**	**(3)**	**(4)**
BK	80	10^6	4,368,112	3,877,966	4,689,762	3,877,966	245,073	0	405,898	0
	70	10^7	37,534,390	29,845,698	34,779,694	29,845,698	3,844,346	0	2,466,998	0
	60	10^7	TO	142,367,230	160,022,752	142,368,486	TO	0	8,827,761	628
	50	10^8	TO	392,514,690	427,960,572	392,752,216	TO	0	17,722,941	118,763
	20	10^8	TO	TO	TO	1,897,581,854	TO	TO	TO	30,869,283
Cancer	95	10^4	90,974	90,536	94,786	90,536	219	0	2,125	0
	94	10^5	259,380	257,200	267,676	257,200	1,090	0	5,238	0
	92	10^5	3,648,522	3,558,590	3,659,250	3,558,716	44,966	0	50,330	63
	90	10^6	15,853,148	15,265,612	15,627,228	15,267,264	293,768	0	180,808	826
AP	80	10^5	923,576	693,480	693,480	693,480	115,048	0	0	0
	70	10^6	7,418,788	5,052,292	5,052,292	5,052,292	1,183,248	0	0	0
	60	10^6	22,198,310	14,735,292	14,735,292	14,735,292	3,731,509	0	0	0
	50	10^7	TO	32,428,688	32,428,688	32,428,688	TO	0	0	0
	20	10^7	TO	133,758,728	133,758,728	133,758,728	TO	0	0	0
	0	10^7	TO	TO	164,934,246	164,934,246	TO	TO	0	0
CH	95	10^6	21,972	18,804	33,494	18,980	1,584	0	7345	88
	90	10^5	701,798	506,656	777,472	531,972	97,571	0	135408	12,658
	85	10^6	6,509,220	3,687,466	5,443,892	3,860,938	1,410,877	0	878,213	86,736
	80	10^6	26,199,798	13,405,308	19,709,578	13,881,322	6,397,245	0	3,152,135	238,007
	50	10^8	TO	TO	TO	TO	TO	TO	TO	TO
LW	80	10^6	3,353,590	2,879,232	6,033,266	2,879,232	237,179	0	1,577,017	0
	70	10^6	25,279,582	20,554,910	40,695,218	20,586,528	2,362,336	0	10,070,154	15,809
	60	10^7	TO	82,141,172	149,604,942	82,995,072	TO	0	33,731,885	426,950
	50	10^8	TO	239,579,812	TO	246,055,584	TO	0	TO	3,237,886
	20	10^8	TO	TO	TO	TO	TO	TO	TO	TO
NT	80	10^3	9,376	7,384	12,300	7,384	996	0	2,458	0
	50	10^4	62,540	47,274	88148	47,274	7,633	0	20,437	0
	20	10^4	226,870	157,224	227,614	161,850	34,823	0	35,195	2,313
	10	10^5	331,282	217,726	279,550	233,534	56,778	0	30,912	7,904
	0	10^5	500,986	255,700	307,964	303,222	122,643	0	26,132	23,761
Yacht	80	10^4	51,494	34,800	160,450	34,808	8,347	0	62,825	4
	50	10^6	8,730,446	2,996,282	8,787,092	2,996,562	2,867,082	0	2,895,405	140
	40	10^6	32,695,988	8,790,198	21,964,420	8,790,706	11,952,895	0	6,587,111	254
	30	10^7	TO	22,182,430	48,031,724	22,183,298	TO	0	12,924,647	434
	20	10^7	TO	48,828,492	93,173,534	48,829,874	TO	0	22,172,521	691
	0	10^7	TO	TO	TO	145,014,502	TO	TO	TO	4,386,766

(1): CP4IM **(2):** CLOSEDPATTERN **(3):** CP4CIP **(4):** GC4CIP

Table 2. Tree searches and failures for the itemsets mining approaches compared to the FCIP mining approaches

Secondly, we compare our global constraint GC4CIP with CLOSEDPATTERN. We observe that the two methods generate tree searches of the same sizes in the AP dataset and in the high frequencies of most datasets (BK, Cancer, LW, NT) with no failures. However, in other instances, CLOSEDPATTERN produces slightly smaller tree searches than GC4CIP (with at most 3% difference between the tree searches). This can be attributed to the backtrack-free nature of the CLOSEDPATTERN approach.

3 Initializing forest algorithm

In this section, we will explain how to initialize our forest structure. Our forest, denoted as F, contains $|\mathcal{M}|$ trees that need to be created before starting the search. For each attribute m in M, we will build a tree starting from the leaves, which will contain a pair of values $(v_{g,m},\ v_{g,m})$ for any $g \in \mathcal{G}$ (line 5 in Algorithm 1). Next, we build the intermediate nodes until reaching the root, by assigning each parent node at most B children. Each parent node will be initialized in its lower bound value with the minimum of the lower bounds of its children and for the upper bound value, it will be set to the maximum of the upper bounds of its children, repeating this process until the root is reached (Line 7 to Line 12).

Algorithm 1: Forest initialization

```
1  Function InitializeForest(N: Database, B: Block-Size)
2      F ⟵ empty list of trees
3      foreach m ∈ M do
4          depth ⟵ ⌈log_B |G|⌉ -1
5          Create |G| tree leaves having (v_{g,m}, v_{g,m}) ∀g∈G
6          NbNodes ⟵ |G|
7          while depth ≠ 0 do
8              NbNodes ⟵ NbNodes /B
9              Create NbNodes parent nodes having:
10             inf(parent) ⟵min(inf(c ∈ child(parent)))
11             sup(parent) ⟵max(sup(c ∈ child(parent)))
12             depth ⟵ depth - 1
13         F ⟵ F ++ Tree
```

4 Detailed K conceptual clustering model using GC4CIP

In this section, we present the detailed model of K-GC4CIP by first describing the variables, then the constraints, and finally the objective function that enables to find the best k-clustering within the search space.

Variables and Domains. Two sets of variables are required to represent an interval pattern associated to a cluster. $\underline{X}^i$ and $\overline{X}^i$ respectively, designate the lower and upper interval borders for a cluster i. These variables are essential for defining the interval borders for the k patterns that characterize our clusters. Initially, their domains contain all the values in the database for each attribute (i.e.$\forall m \in \mathcal{M}, \underline{x}^i_m \in \underline{X}^i, \overline{x}^i_m \in \overline{X}^i$, $\underline{x}^i_m = \overline{x}^i_m =$

$\mathcal{N}_m$). Additionally, this model requires another set of variables for each cluster, denoted as Y^i, to indicate object cover.The domain of the variable y_g^i contains initially the values 0 and 1, where 0 signifies that the object g is not part of cluster i, while 1 indicates that the object g is included in cluster i.

Constraints. To find a k-clustering, our model requires the conjunction of two distinct types of constraints. The first constraint is a closure constraint, represented by our global constraint $\text{GC4CIP}_{\mathcal{N},\theta}(\underline{X}^i, \overline{X}^i, Y^i)$. By applying this constraint k times, we generate k closed interval patterns, resulting in the formation of a k clustering. Furthermore, to ensure the non-overlapping coverage of our interval patterns, we introduce a partitioning constraint. This constraint 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. $\sum_{1\leq i\leq k} y_g^i = 1$).

Objective function. To extract the most interesting clustering from the search space based on the Euclidean distance metric $d(i,j) = \sqrt{\sum_{m=1}^{|\mathcal{M}|} \left(v_{j,m} - v_{j,m}\right)^2}$, we define an objective function designed to select the solution that minimizes the cumulative intra-cluster distance D^k among our clusters. D^k correspond to the sum of the Euclidean distances in a cluster k (i.e. $\sum_{1\leq|i|\leq|j|\leq|\mathcal{G}|} d_{i,j} \, . \, y_i^k \, . \, y_j^k = D^k$). Our objective function is denoted as $min \sum_{1\leq i\leq k} D^i$.