Supplementary Material

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

University of Caen Normandy, Caen 14000, France
first-name.last-name@unicaen.fr

1 Coverage based filtering rules

	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 $\mathcal{N}$

Let $g, g', g'' \in \mathcal{G}$. $\forall m \in \mathcal{M}$.

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

Proof. Let the partial assignement $\mathcal{V}^* = \langle [min(\underline{x}_1), max(\overline{x}_1)], ...[min(\underline{x}_{|\mathcal{M}|}), max(\overline{x}_{|\mathcal{M}|})] \rangle$ and consider $g, g', g'' \in \mathcal{G}$ 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. $\mathcal{D}(y_g) = \{0, 1\}$.
Assume that $1 \notin \mathcal{D}(y_g)$ 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 1. 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 1, the value 0 will be filtered from $\mathcal{D}(y_2)$ and $\mathcal{D}(y_4)$ since $\forall m \in \mathcal{M}$ $v_{2,m}$, $v_{4,m} \in \langle [2, 6], [7, 12], [91, 130] \rangle$.

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

$$1 \notin \mathcal{D}(y_g) \text{ if } \begin{cases} \mathcal{D}(y_{g'}) = \{0\} \wedge \mathcal{D}(y_{g''}) = \{1\} \\ \wedge \\ \forall m \in \mathcal{M},\ v_{g,m} \leq v_{g',m} < v_{g'',m} \ \vee\ v_{g'',m} < v_{g',m} \leq v_{g,m} \end{cases} \tag{2}$$

Proof. We prove that if there exist $g',\ g'' \in \mathcal{G}$ where $\mathcal{D}(y_{g'}) = \{0\},\ \mathcal{D}(y_{g''}) = \{1\}$ and for all $m \in \mathcal{M}\ v_{g',m} < v_{g'',m} \vee v_{g'',m} < v_{g',m}$ (total order relation), for a given non instantiated object $g \in \mathcal{G}$ (i.e. $\mathcal{D}(y_g) = \{0,1\}$) where for all $m \in \mathcal{M}\ v_{g,m} \leq v_{g',m} < v_{g'',m} \vee v_{g'',m} < v_{g',m}$ (total order relation) g must be covered.

Suppose that $g, g', g'' \in \mathcal{G}$ where $\mathcal{D}(y_g) = \{1\},\ \mathcal{D}(y_{g'}) = \{0\},\ \mathcal{D}(y_{g''}) = \{1\}$ and there is a total order relation between g, g', g'' such that, for all $m \in \mathcal{M}\ v_{g,m} \leq v_{g',m} < v_{g'',m}$ or $v_{g'',m} < v_{g',m}$. $\mathcal{D}(y_g) = \{1\}$ and $\mathcal{D}(y_{g''}) = \{1\}$ mean that there exist an interval pattern $\mathcal{V}$ in which g, g" are occurrences. Following the FCIP definition in the paper $\underline{x}_m = min(\{v_{g,m} \mid g \in cover(\mathcal{V})\})$, g' can not be an occurence of $\mathcal{V}$ as $\mathcal{D}(y_{g'}) = \{0\}$. Therefore, having a total order relation between g, g', g'', g can not ve an occurence of $\mathcal{V}$ as the values occuring in g are smaller resp. greater than the values occuring in g'. Thus $1 \notin \mathcal{D}(y_g)$ which contradict the assumption.

Example 2. 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 2, 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 our FCIP mining model (CP4CIP) with 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. Lastly, in the LW database, CP4IM generate a tree that is 70% smaller than CP4CIP. Secondly, we compare our global constraint GC4CIP to 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.

$\mathcal{N}$	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

2.1 Modelling a k-clustering problem using GC4CIP

In this section, 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 data set as akin to K-MEANS method. Since we use closed interval pattern to form each cluster, this task is known as conceptual clustering where each cluster is described by a unique concept (i.e. closure property). This task of finding k interval 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 model this task we can use our global constraint GC4CIP to carry the conceptual clustering directly from the numerical data. This involves a set of variables and constraints described in the following:

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^i_g 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. $\sum_{1 \leq i \leq k} y^i_g = 1$).

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. $\sum_{1 \leq |i| \leq |j| \leq |\mathcal{G}|} d_{i,j} \,.\, y^k_i \,.\, y^k_j = D^k$). Our objective function is denoted as $min \sum_{1 \leq i \leq k} D^i$.