



HAL
open science

A Global Constraint for Closed Frequent Pattern Mining

Nadjib Lazaar, Yahia Lebbah, Samir Loudni, Mehdi Maamar, Valentin Lemière, Christian Bessiere, Patrice Boizumault

► **To cite this version:**

Nadjib Lazaar, Yahia Lebbah, Samir Loudni, Mehdi Maamar, Valentin Lemière, et al.. A Global Constraint for Closed Frequent Pattern Mining. CP 2016 - 22nd International Conference on Principles and Practice of Constraint Programming, Sep 2016, Toulouse, France. pp.333-349, 10.1007/978-3-319-44953-1_22 . lirmm-01374719

HAL Id: lirmm-01374719

<https://hal-lirmm.ccsd.cnrs.fr/lirmm-01374719v1>

Submitted on 15 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Global Constraint for Closed Frequent Pattern Mining

N. Lazaar¹, Y. Lebbah², S. Loudni³, M. Maamar^{1,2},
V. Lemièrè³, C. Bessiere¹, P. Boizumault³

¹ LIRMM, University of Montpellier, France

² LITIO, University of Oran 1 Ahmed Ben Bella, Algeria

³ GREYC, Normandie University, Caen, France

Abstract. Discovering the set of closed frequent patterns is one of the fundamental problems in Data Mining. Recent Constraint Programming (CP) approaches for declarative itemset mining have proven their usefulness and flexibility. But the wide use of reified constraints in current CP approaches leads to difficulties in coping with high dimensional datasets. In this paper, we propose the CLOSEDPATTERN global constraint to capture the closed frequent pattern mining problem without requiring reified constraints or extra variables. We present an algorithm to enforce *domain consistency* on CLOSEDPATTERN in polynomial time. The computational properties of this algorithm are analyzed and its practical effectiveness is experimentally evaluated.

1 Introduction

Frequent Pattern Mining is a well-known and perhaps the most popular research field of data mining. Originally introduced by Agrawal et al. [1], it plays a key role in many data mining applications. These applications include the discovery of frequent itemsets and association rules [1], correlations [2] and many other data mining tasks.

In practice, the number of frequent patterns produced is often huge and can easily exceed the size of the input dataset. Most frequent patterns are redundant and can be derived from other found patterns. Hence, *closed frequent patterns* have been introduced. They provide a concise and condensed representation that avoids redundancy. Discovering the set of closed frequent patterns is one of the fundamental problems in Data Mining. Several specialized approaches have been proposed to discover closed frequent patterns (e.g., A-CLOSE algorithm [12], CHARM [17], CLOSET [13], LCM [14]).

Over the last decade, the use of the Constraint Programming paradigm (CP) to model and to solve Data Mining problems has received considerable attention [3,5,9]. The declarative aspect represents the key success of the proposed CP approaches. Doing so, one can add/remove any user-constraint without the need of developing specialized solving methods.

Related to the Closed Frequent Pattern Mining problem (CFPM), Guns et al., propose to express the different constraints that we can have in Pattern Mining as a CP model [5]. The model is expressed on Boolean variables representing

items and transactions, with a set of reified sum constraints. The reified model has become a de facto standard for many DM tasks. Indeed, the reified model had been adopted for the k -pattern sets [5]. The drawback is the wide use of reified constraints in the CP model, which makes the scalability of the approach questionable.

In the line of the work of Kemmar et al. [8], we propose in this paper the CLOSEDPATTERN global constraint. CLOSEDPATTERN does not require reified constraints and extra variables to encode and propagate the CFPM problem. CLOSEDPATTERN captures the particular semantics of the CFPM problem and domain consistency can be achieved on it using a polynomial algorithm. Experiments on several known large datasets show that our approach outperforms the reified model used in CP4IM [3] and is more scalable, which is a major issue for CP approaches. This result can be explained by the fact that CLOSEDPATTERN insures domain consistency.

The paper is organized as follows. Section 2 recalls preliminaries. Section 3 provides the context and the motivations for the CLOSEDPATTERN global constraint. Section 4 presents the global constraint CLOSEDPATTERN. Section 5 illustrates the power of the pruning algorithm compared with the reified model. Section 6 reports experiments. Finally, we conclude and draw some perspectives.

2 Background

In this section, we introduce some useful notions used in closed frequent pattern mining and constraint programming.

2.1 Closed frequent pattern mining

Let $\mathcal{I} = \{1, \dots, n\}$ be a set of n item indices¹ and $\mathcal{T} = \{1, \dots, m\}$ a set of m transaction indices. A pattern P (i.e., itemset) is a subset of \mathcal{I} . The language of patterns corresponds to $\mathcal{L}_{\mathcal{I}} = 2^{\mathcal{I}}$. A transaction database is a set $\mathcal{D} \subseteq \mathcal{I} \times \mathcal{T}$. The set of items corresponding to a transaction identified by t is denoted by $\mathcal{D}[t] = \{i \mid (i, t) \in \mathcal{D}\}$. A transaction t is an occurrence of some pattern P iff the set $\mathcal{D}[t]$ contains P (i.e., $P \subseteq \mathcal{D}[t]$).

The cover of P , denoted by $\mathcal{T}_{\mathcal{D}}(P)$, is the set of transactions containing P , that is, $\mathcal{T}_{\mathcal{D}}(P) = \{t \in \mathcal{T} \mid P \subseteq \mathcal{D}[t]\}$. Given $S \subseteq \mathcal{T}$ a subset of transactions, $\mathcal{I}_{\mathcal{D}}(S)$ is the set of common items of S , that is, $\mathcal{I}_{\mathcal{D}}(S) = \bigcap_{t \in S} \mathcal{D}[t]$. The (absolute) frequency of a pattern P is the size of its cover (i.e., $freq_{\mathcal{D}}(P) = |\mathcal{T}_{\mathcal{D}}(P)|$). Let $\theta \in \mathbb{N}^+$ be some given constant called a *minimum support*. A pattern P is frequent if $freq_{\mathcal{D}}(P) \geq \theta$.

Example 1. Consider the transaction database in Table 1. We have $\mathcal{T}_{\mathcal{D}}(C) = \{t_1, t_3\}$, $freq_{\mathcal{D}}(C) = 2$ and $\mathcal{I}_{\mathcal{D}}(\{t_1, t_3\}) = CH$.

¹ For the sake of readability, our examples refer to items by their names instead of their indices.

Table 1: A transaction database \mathcal{D} (a) and its binary matrix (b).

(a)						(b)								
Trans.	Items					Trans.	A	B	C	D	E	F	G	H
t_1	B	C			G H	t_1	0	1	1	0	0	0	1	1
t_2	A		D			t_2	1	0	0	1	0	0	0	0
t_3	A	C	D		H	t_3	1	0	1	1	0	0	0	1
t_4	A			E F		t_4	1	0	0	0	1	1	0	0
t_5	B		E F G			t_5	0	1	0	0	1	1	1	0

The closure of a pattern P in \mathcal{D} , denoted by $Clos(P)$, is the set of common items of its cover $\mathcal{T}_{\mathcal{D}}(P)$, that is, $Clos(P) = \mathcal{I}_{\mathcal{D}}(\mathcal{T}_{\mathcal{D}}(P))$. A pattern is closed if and only if $Clos(P) = P$.

Definition 1 (Closed Frequent Pattern Mining (CFPM)). *Given a transaction database \mathcal{D} and a minimum support threshold θ , the closed frequent pattern mining problem is the problem of finding all patterns P such that $(freq_{\mathcal{D}}(P) \geq \theta)$ and $(Clos(P) = P)$.*

Example 2. For $\theta = 2$, the set of closed frequent patterns in Table 1 is $\emptyset\langle 5 \rangle$,² $A\langle 3 \rangle$, $AD\langle 2 \rangle$, $BG\langle 2 \rangle$, $CH\langle 2 \rangle$ and $EF\langle 2 \rangle$.

Closed frequent patterns provide a minimal representation of frequent patterns, i.e. we can derive all frequent patterns with their exact frequency value from the closed ones [12]. We now define the important notion of *full extension* that comes from pattern mining algorithms and that we will use later in this paper.

Definition 2 (Full extension). *The non-empty itemset Q is called a full extension of P iff $\mathcal{T}_{\mathcal{D}}(P) = \mathcal{T}_{\mathcal{D}}(P \cup Q)$.*

Definition 2 is at the key of the item merging property [15] stated as follows: If some pattern Q is a full extension of some pattern P , and none of the proper supersets of Q is a full extension of P , then $P \cup Q$ forms a closed pattern. In other words, a closed pattern can be defined as a pattern that does not possess a full extension.

Search Space Issues. In pattern mining, the search space contains $2^{\mathcal{I}}$ candidates. Given a large number of items \mathcal{I} , a naive search that consists of enumerating and testing the frequency of pattern candidates in a dataset is infeasible. The main property exploited by most algorithms to reduce the search space is that frequency is *monotone decreasing* with respect to extension of a set.

Property 1 (Anti-monotonicity of the frequency). Given a transaction database \mathcal{D} over \mathcal{I} , and two patterns $X, Y \subseteq \mathcal{I}$. Then, $X \subseteq Y \Rightarrow freq_{\mathcal{D}}(Y) \leq freq_{\mathcal{D}}(X)$.

Hence, any subset (resp. superset) of a frequent (resp. infrequent) pattern is also a frequent (resp. infrequent) pattern.

² Value between $\langle \cdot \rangle$ indicates the frequency of a pattern.

2.2 CFPM under constraints

Constraint-based pattern mining aims at extracting all patterns P of $\mathcal{L}_{\mathcal{I}}$ satisfying a query $q(P)$ (conjunction of constraints), which usually defines what we call a *theory* [10]: $Th(q) = \{P \in \mathcal{L}_{\mathcal{I}} \mid q(P) \text{ is true}\}$. A common example is the frequency measure leading to the frequent pattern constraint. It is also possible to have other kind of (user-)constraints. For instance, constraints on the size of the returned patterns, $minSize(P, \ell_{min})$ constraint holds if and only if the number of items of P is greater than or equal to ℓ_{min} . Constraints on the presence of an item in a pattern $item(P, i)$ state that an item i must be in a pattern P .

2.3 CSP and global constraints

A constraint network is defined by a set of variables $X = \{x_1, \dots, x_n\}$, each variable $x_i \in X$ having an associated finite domain $dom(x_i)$ of possible values, and a set of constraints \mathcal{C} on X . A constraint $c \in \mathcal{C}$ is a relation that specifies the allowed combinations of values for its variables $X(c)$. An assignment σ is a mapping from variables in X to values in their domains. The *Constraint Satisfaction Problem* (CSP) consists in finding an assignment satisfying all constraints.

Domain Consistency (DC). Constraint solvers typically use backtracking search to explore the search space of partial assignments. At each assignment, filtering algorithms prune the search space by enforcing local consistency properties like domain consistency. A constraint c on $X(c)$ is domain consistent, if and only if, for every $x_i \in X(c)$ and every $d_i \in dom(x_i)$, there is an assignment σ satisfying c such that $x_i = d_i$.

Global constraints are constraints capturing a relation between a non-fixed number of variables. These constraints provide the solver with a better view of the structure of the problem. Examples of global constraints are AllDifferent, Regular and Among (see [7]). Except the case when for a given global constraint a Berge-acyclic decomposition exists, global constraints cannot be efficiently propagated by generic local consistency algorithms, which are exponential in the number of the variables of the constraint. Dedicated filtering algorithms are constructed to achieve polynomial time complexity in the size of the input, i.e., the domains and extra parameters. The aim of this paper is to propose a filtering algorithm for the frequent closed pattern constraint.

3 Context and Motivations

This section provides a critical review of ad-hoc specialized methods and CP approaches for CFPM, and motivates the proposition of a global constraint.

Specialized methods for CFPM. CLOSE [12] was the first algorithm proposed to extract closed frequent patterns (CFPs). It uses an apriori-like bottom-up method. Later, Zaki and Hsiao [17] proposed a depth-first algorithm based on a vertical database format e.g. CHARM. In [13], Pei et al. extended the FP-growth method to a method called CLOSET for mining CFPs. Lastly, Uno et al. [14] have

proposed LCM, one of the fastest frequent itemset mining algorithm. It uses a hybrid representation based on vertical and horizontal representations. The milestone of LCM is a technique called *prefix preserving closure extension* (PPCE), which allows to generate a new frequent closed pattern from a previously obtained closed pattern. Let us explain the PPCE principle. Consider the closed pattern P . Let $P(i) = P \cap \{1, \dots, i\}$ be the subset of P consisting of items no greater than i . The core index of P , denoted by $core(P)$, is the minimum index i such that $\mathcal{T}_{\mathcal{D}}(P(i)) = \mathcal{T}_{\mathcal{D}}(P)$. A pattern Q is PPCE of P if $Q = Clos(P \cup \{i\})$ and $P(i-1) = Q(i-1)$ for an item $i \notin P$ and $i > core(P)$. The completeness of PPCE is guaranteed by the following property: If Q is a nonempty closed itemset, then there is only one closed itemset P such that Q is a PPCE of P . The mining process using LCM is a depth first search where at each node we have a closed pattern P . LCM uses the PPCE technique as a branching strategy to jump from a closed pattern to other closed patterns by adding new items. With its specialized depth first search, LCM succeeds to enumerate very quickly the closed patterns. However, if the user considers other (user-)constraints on patterns, the search procedure should be revised. In fact, all these specialized proposals (e.g., Closet, Charm, LCM, etc.), though efficient, are ad-hoc methods suffering from the lack of genericity, since adding new constraints requires new implementations.

Reified constraint model for itemset mining. De Raedt et al. have proposed in [3] a CP model for itemset mining. They show how some constraints (e.g., frequency, maximality, closedness) can be modeled as CSP [11,6]. This modeling uses two sets of Boolean variables P and T : (1) Decision variables: item variables P_1, P_2, \dots, P_n , where $P_i = 1$ if and only if item i is in the searched pattern; (2) Auxiliary variables: transaction variables T_1, T_2, \dots, T_m , where $T_t = 1$ if and only if the searched pattern is in $\mathcal{D}[t]$.

The relationship between P and T , set of channeling constraints, is modeled by reified constraints stating that, for each transaction t , $(T_t = 1)$ iff P is a subset of $\mathcal{D}[t]$: $\forall t \in \mathcal{T} : (T_t = 1) \leftrightarrow \sum_{i \in \mathcal{I}} P_i(1 - \mathcal{D}[t, i]) = 0$ (arity $n + 1$). The min frequency constraint is modeled us: $\forall i \in \mathcal{I} : (P_i = 1) \rightarrow \sum_{t \in \mathcal{T}} T_t \mathcal{D}[t, i] \geq \theta$ (arity $m + 1$). The closedness constraint is expressed with: $\forall i \in \mathcal{I} : (P_i = 1) \leftrightarrow \sum_{t \in \mathcal{T}} T_t(1 - \mathcal{D}[t, i]) = 0$ (arity $m + 1$). Such encoding has a major drawback since it requires $(m + n + n)$ reified constraints of arity $(n + 1)$ and $(m + 1)$ to encode the whole database. This constitutes a strong limitation especially when it comes to handle very large databases.

We propose in the next section the CLOSEDPATTERN global constraint to encode both the minimum frequency constraint and the closedness constraint. This global constraint requires neither reified constraints nor auxiliary variables.

4 CLOSEDPATTERN Constraint

This section presents the CLOSEDPATTERN global constraint for the CFPM problem.

4.1 Definition and filtering

Let P be the unknown pattern we are looking for. The unknown pattern P is encoded with Boolean item variables P_1, \dots, P_n . In the rest of the paper we will denote by σ the partial assignment obtained from the variables P_1, \dots, P_n that have a singleton domain. We will also use the following subsets of items:

- present items: $\sigma^+ = \{j \in 1..n \mid P_j = 1\}$,
- absent items: $\sigma^- = \{j \in 1..n \mid P_j = 0\}$,
- other items: $\sigma^* = \{1..n\} \setminus (\sigma^+ \cup \sigma^-)$.

σ^* is the set of free items (non instantiated variables). If $\sigma^* = \emptyset$ then σ is a complete assignment.

The global constraint CLOSEDPATTERN ensures both the minimum frequency property and the closedness property.

Definition 3 (CLOSEDPATTERN global constraint). *Let P_1, \dots, P_n be binary item variables. Let \mathcal{D} be a transaction database and θ a minimum support. Given a complete assignment σ on P_1, \dots, P_n , $\text{CLOSEDPATTERN}_{\mathcal{D},\theta}(\sigma)$ holds if and only if $\text{freq}_{\mathcal{D}}(\sigma^+) \geq \theta$ and σ^+ is closed.*

Example 3. Consider the transaction database of Table 1a with $\theta = 2$. Let $P = \langle P_1, \dots, P_8 \rangle$ with $\text{dom}(P_i) = \{0, 1\}$ for $i \in 1..8$. Consider the closed pattern AD encoded by $P = \langle 10010000 \rangle$, where $\sigma^+ = \{A, D\}$ and $\sigma^- = \{B, C, E, F, G, H\}$. $\text{CLOSEDPATTERN}_{\mathcal{D},2}(P)$ holds because $\text{freq}_{\mathcal{D}}(\{A, D\}) \geq 2$ and $\{A, D\}$ is closed.

Let σ be a partial assignment of variables P and i a free item. We use the vertical representation of the dataset, denoted $\mathcal{V}_{\mathcal{D}}$ where for each item, the transactions containing it are stored: $\forall i \in \mathcal{I}, \mathcal{V}_{\mathcal{D}}(i) = \mathcal{T}_{\mathcal{D}}(\{i\})$. We denote by $\mathcal{V}_{\mathcal{D}}^{\sigma^+}(i)$ the cover of item i within the current cover of a pattern σ^+ :

$$\mathcal{V}_{\mathcal{D}}^{\sigma^+}(i) = \mathcal{T}_{\mathcal{D}}(\sigma^+ \cup \{i\}) = \mathcal{T}_{\mathcal{D}}(\sigma^+) \cap \mathcal{T}_{\mathcal{D}}(\{i\}).$$

We need to define extensible assignments.

Definition 4 (Extensible assignment). *Given a constraint $\text{CLOSEDPATTERN}_{\mathcal{D},\theta}$ on P_1, \dots, P_n , a partial assignment is said to be extensible if and only if it can be extended to a complete assignment of P_1, \dots, P_n that satisfies $\text{CLOSEDPATTERN}_{\mathcal{D},\theta}$.*

We show when a partial assignment is extensible with respect to CLOSED-PATTERN constraint.

Proposition 1. *Let σ be a partial assignment of variables in P_1, \dots, P_n . σ is an extensible partial assignment if and only if $\text{freq}_{\mathcal{D}}(\sigma^+) \geq \theta$ and $\nexists j \in \sigma^-$ such that $\{j\}$ is a full extension of σ .*

Proof. According to the anti-monotonicity property of the frequency (cf. Property 1), if the partial assignment σ is infrequent (i.e., $\text{freq}_{\mathcal{D}}(\sigma^+) < \theta$), it cannot, under any circumstances, be extended to a closed pattern.

Given now a frequent partial assignment σ (i.e., $\text{freq}_{\mathcal{D}}(\sigma^+) \geq \theta$), let us take $j \in \sigma^-$ such that $\{j\}$ is a full extension of σ . It follows that $\mathcal{T}_{\mathcal{D}}(\sigma^+) = \mathcal{T}_{\mathcal{D}}(\sigma^+ \cup \{j\}) = \mathcal{V}_{\mathcal{D}}^{\sigma^+}(j)$. Therefore, $\text{Clos}(\sigma^+) = \text{Clos}(\sigma^+ \cup \{j\})$. Since σ^+ without j (j being in σ^-) cannot be extended to a closed pattern, the result follows. If there is no item $j \in \sigma^-$ such that $\{j\}$ is a full extension of σ , then the current assignment σ can be definitely extended to a closed itemset by adopting a full extension to form a closed pattern. \square

We now give the CLOSEDPATTERN filtering rules by showing when a value of a given variable is inconsistent.

Proposition 2 (CLOSEDPATTERN filtering rules). *Let σ be an extensible partial assignment of variables in P_1, \dots, P_n , and P_j ($j \in \sigma^*$) be a free variable. The following two cases characterize the inconsistency of the values 0 and 1 of P_j :*

- $0 \notin \text{dom}(P_j)$ iff: $\{j\}$ is a full extension of σ . (rule 1)
- $1 \notin \text{dom}(P_j)$ iff: $\begin{cases} |\mathcal{V}_{\mathcal{D}}^{\sigma^+}(j)| < \theta & \vee \\ \exists k \in \sigma^-, \mathcal{V}_{\mathcal{D}}^{\sigma^+}(j) \subseteq \mathcal{V}_{\mathcal{D}}^{\sigma^+}(k). \end{cases}$ (rule 2)
(rule 3)

Proof. Let σ be an extensible partial assignment and P_j be a free variable.

- $0 \notin \text{dom}(P_j)$: (\Rightarrow) Let 0 be an inconsistent value. In this case, P_j can only take value 1. It means that $\text{Clos}(\sigma^+) = \text{Clos}(\sigma^+ \cup \{j\})$. Thus, $\mathcal{T}_{\mathcal{D}}(\sigma^+) = \mathcal{T}_{\mathcal{D}}(\sigma^+ \cup \{j\})$. By definition 2, $\{j\}$ is a full extension of σ .
 (\Leftarrow) Let $\{j\}$ be a full extension of σ , which means that $\text{Clos}(\sigma^+) = \text{Clos}(\sigma^+ \cup \{j\})$ (def. 2). The value 0 is inconsistent where j cannot be in σ^- (prop.1).
- $1 \notin \text{dom}(P_j)$: (\Rightarrow) Let 1 be an inconsistent value. This can be the case if the frequency of the current pattern σ^+ is set up below the threshold θ by adding the item j (i.e., $|\mathcal{V}_{\mathcal{D}}^{\sigma^+}(j)| < \theta$). Or, $\sigma^+ \cup \{j\}$ cannot be extended to a closed itemset: this is the case when there exists an item $k \in \sigma^-$ such that at each time the item j belongs to a transaction in the database, k belongs as well ($\mathcal{V}_{\mathcal{D}}^{\sigma^+}(j) \subseteq \mathcal{V}_{\mathcal{D}}^{\sigma^+}(k)$). Conversely, the lack of k (i.e., $k \in \sigma^-$) implies the lack of j as well. This means that: $(P_k = 0 \Rightarrow P_j = 0)$.
 (\Leftarrow) This is a direct consequence of Proposition 1. \square

The first rule takes its origin from item merging [15]. The second rule is a basic rule derived from the property of anti-monotonicity of the frequency (Property 1). To the best of our knowledge, the third rule is a new rule taking its originality from the reasoning made on absent items.

Example 4. Following Example 3, consider a partial assignment σ such that the variable P_1 is set to 0 (item A). That is, $\sigma^- = \{A\}$ and $\sigma^+ = \emptyset$. Value 1 from $\text{dom}(P_4)$ (item D) is inconsistent because the lack of A implies the lack of D in \mathcal{D} (i.e., $\mathcal{V}_{\mathcal{D}}^{\sigma^+}(D) \subseteq \mathcal{V}_{\mathcal{D}}^{\sigma^+}(A)$). Let now $P_1 = 0, P_4 = 0$, that is, $\sigma^- = \{A, D\}$ and $\sigma^+ = \emptyset$. If the variable P_3 is set to 1 (item C), value 1 from $P_i, i = \{2, 5, 6, 7\}$ (items B, E, F, G) is inconsistent because $|\mathcal{V}_{\mathcal{D}}^{\sigma^+}(P_i)| < 2$, and value 0 from P_8 (item H) is also inconsistent because $\{H\}$ is a full extension of σ .

Algorithm 1: FILTER-CLOSEDPATTERN($\mathcal{V}_D, \theta, \sigma, P$)

```
1 Input:  $\mathcal{V}_D$  : vertical database;  $\theta$  : minimum support
2 InOut:  $P = \{P_1 \dots P_n\}$ : Boolean item variables;  $\sigma$  : current assignment.
3 begin
4 if ( $|\mathcal{T}_D(\sigma^+)| < \theta$ ) then return false;
5 if  $\exists i \in \sigma^- : |\mathcal{V}_D^{\sigma^+}(i)| = |\mathcal{T}_D(\sigma^+)|$  then return false;
6 foreach  $i \in \sigma^*$  do
7   if ( $|\mathcal{V}_D^{\sigma^+}(i)| = |\mathcal{T}_D(\sigma^+)|$ ) then
8      $dom(P_i) \leftarrow dom(P_i) - \{0\}$ ;
9      $\sigma^+ \leftarrow \sigma^+ \cup \{i\}$ ;  $\sigma^* \leftarrow \sigma^* \setminus \{i\}$ ;
10  else if ( $|\mathcal{V}_D^{\sigma^+}(i)| < \theta$ ) then
11     $dom(P_i) \leftarrow dom(P_i) - \{1\}$ ;
12     $\sigma^- \leftarrow \sigma^- \cup \{i\}$ ;  $\sigma^* \leftarrow \sigma^* \setminus \{i\}$ ;
13 foreach  $i \in \sigma^-$  do
14   foreach  $j \in \sigma^* : \mathcal{V}_D^{\sigma^+}(j) \subseteq \mathcal{V}_D^{\sigma^+}(i)$  do
15      $dom(P_j) \leftarrow dom(P_j) - \{1\}$ ;
16      $\sigma^- \leftarrow \sigma^- \cup \{j\}$ ;  $\sigma^* \leftarrow \sigma^* \setminus \{j\}$ ;
17 return true;
```

4.2 CLOSED PATTERN Filtering Algorithm

In this section, we present the algorithm FILTER-CLOSEDPATTERN (Algorithm 1) for enforcing domain consistency on the CLOSED PATTERN constraint. FILTER-CLOSEDPATTERN incrementally maintains the internal data structures $\sigma = \langle \sigma^+, \sigma^-, \sigma^* \rangle$ and the corresponding cover $\mathcal{T}_D(\sigma^+)$. Using these two structures, one can check if an item is present or not in the vertical dataset \mathcal{V}_D .

Algorithm 1 takes as input the vertical dataset \mathcal{V}_D , a minimum support threshold θ , the current partial assignment σ on P where $\sigma^* \neq \emptyset$, and the variables P . As output, Algorithm 1 reduces the domains of P_i 's and therefore, increases σ^+ and/or σ^- , and decreases σ^* .

The algorithm starts by checking if the current partial assignment is extensible or not (Proposition 1). This is performed by checking 1) if the size of the current cover is greater than the minimum support (line 4) and 2) if no item of a variable already instantiated to zero is a full extension of σ^+ (line 5).

Lines 6-12 are a straightforward application of rules 1 and 2 of Proposition 2. For each non-instantiated variable, 1) we check if value 0 is consistent: that item is not a full extension (lines 6-9), and 2) we check if value 1 is consistent: the new cover size by adding that item remains greater than θ (lines 10-12).

Finally, lines 13-16 implement rule 3 of Proposition 2. We prune value 1 from each free item variable $i \in \sigma^*$ such that its cover is a superset of the cover of an absent item $j \in \sigma^-$ ($\mathcal{V}_D^{\sigma^+}(i) \subseteq \mathcal{V}_D^{\sigma^+}(j)$).

Theorem 1. *Given a transaction database \mathcal{D} of n items and m transactions, and a threshold θ . Algorithm FILTER-CLOSEDPATTERN enforces domain*

consistency on the CLOSEDPATTERN constraint, or proves that it is inconsistent in time $O(n^2 \times m)$ with a space complexity of $O(n \times m)$.

Proof. **DC:** FILTER-CLOSEDPATTERN implements exactly Proposition 1 and the three rules given in Proposition 2. Thus FILTER-CLOSEDPATTERN ensures domain consistency (see the description of Algorithm 1).

Time: Let $n = |\mathcal{I}|$ and $m = |\mathcal{T}|$. First, we need to compute $\mathcal{T}_{\mathcal{D}}(\sigma^+)$ which requires at most $O(n \times m)$. This is done only once. The cover $\mathcal{V}_{\mathcal{D}}^{\sigma^+}(i)$ can be computed by intersecting $\mathcal{T}_{\mathcal{D}}(\sigma^+)$ (already computed) and $\mathcal{T}_{\mathcal{D}}(\{i\})$ (given by the vertical representation) within at most $O(m)$. Checking rules 1 and 2 on all free variables can be done in $O(n \times m)$ (lines 6-12). However, checking rule 3 is cubic at lines 13-16 (i.e., $O(n \times (n \times m))$), where checking if a cover $\mathcal{V}_{\mathcal{D}}^{\sigma^+}(i)$ is a subset of another cover can be done in $O(m)$. Finally, the worst case complexity is $O(n \times (n \times m))$.

Space: The space complexity of FILTER-CLOSEDPATTERN lies in the storage of $\mathcal{V}_{\mathcal{D}}$, σ and the cover \mathcal{T} data structures. The vertical representation $\mathcal{V}_{\mathcal{D}}$ requires at most $n \times m$ space. In the worst case, we have to store n items within σ and m transactions within \mathcal{T} . That is, the worst case space complexity is $O(n \times m + n + m) = O(n \times m)$. \square

During the solving process in depth first search, the whole space complexity is $O(n \times (m + n))$ because (1) the depth is at most n ; (2) σ and \mathcal{T} require $O(n \times (m + n))$; (3) the vertical representation is the same data used all along the solving process $O(n \times m)$; (4) $O(n \times (m + n)) + O(n \times m) = O(n \times (m + n))$.

Proposition 3 (Backtrack-free). *Extracting the total number of closed frequent patterns, noted C , is backtrack-free with a complexity in $O(C \times n^2 \times m)$ using FILTER-CLOSEDPATTERN to propagate the CLOSEDPATTERN constraint.*

Proof. FILTER-CLOSEDPATTERN ensures DC at each node of the search tree. Hence, the closed frequent patterns are guaranteed to be produced in a backtrack-free manner. The explored search tree is a binary full tree where each node is either a leaf (a solution) or possesses exactly two child nodes. The number of nodes is thus in $O(2 \times C)$. Knowing that ensuring DC is in $O(n^2 \times m)$, extracting the total number of closed frequent patterns is in $O(C \times n^2 \times m)$. \square

4.3 Data structures

To represent the transactional dataset and the cover of items, we adopted the vertical representation format [16]. Our implementation is in `or-tools` solver ³.

Static structures: for each item $i \in \mathcal{I}$, $\mathcal{T}_{\mathcal{D}}(i) = \{t \in \mathcal{T} \mid i \in t\}$ is stored as a bitset of size m . If $t \in \mathcal{T}_{\mathcal{D}}(i)$, then the associated bit is set to 1 (0 otherwise).

Dynamic structure: a vector *Memo* of bitsets is used to store the cover of each partial solution. *Memo* is a vector of size $n + 1$, since at the beginning of the search, the partial assignment is empty. Let σ be a partial assignment. Each time a (new) variable P_i is instantiated, the cover of the new partial assignment

³ <https://developers.google.com/optimization/>

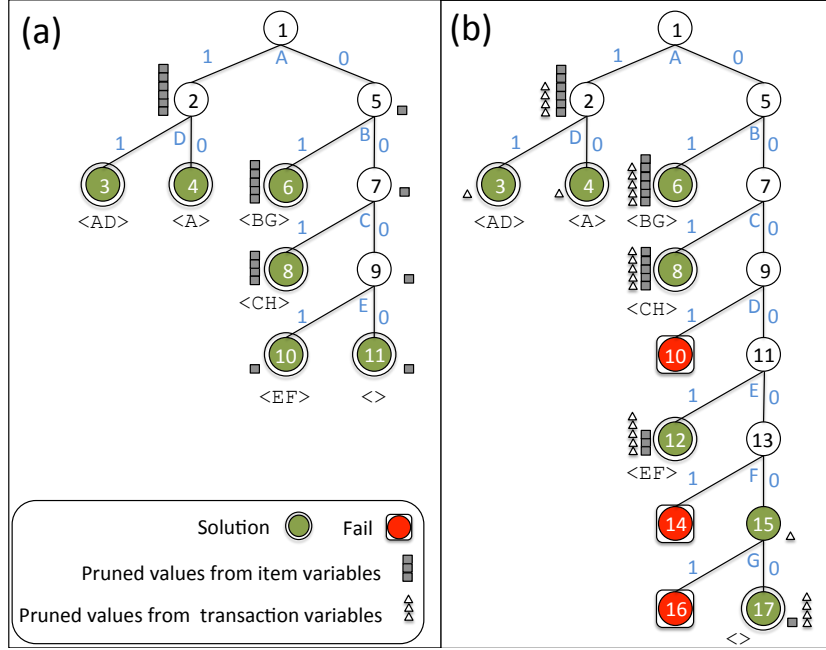


Fig. 1: (a) CLOSEDPATTERN and (b) Reified Constraint Model (RCM)

$\sigma \cup \{i\}$ is stored in *Memo*. If $P_i = 0$, the cover remains the same: $\mathcal{T}_{\mathcal{D}}(\sigma^+ \cup \{i\}) = \mathcal{T}_{\mathcal{D}}(\sigma^+)$. If $P_i = 1$, the cover of the new partial solution $\sigma \cup \{i\}$ is computed by a bitwise-AND between $\mathcal{T}_{\mathcal{D}}(\sigma^+)$ and $\mathcal{T}_{\mathcal{D}}(\{i\})$, and stored in *Memo*.

Backtracking. First, all P_i , as well as their domains $dom(P_i)$, are fully maintained by the `or-tools` backtracking. Then, a single value (the current index of the vector *Memo*) is asked to be managed by the `or-tools` backtracking. Each time a partial solution is extended (from σ to $\sigma \cup \{i\}$), the current index of *Memo* is memorised. When a backtrack occurs (from $\sigma \cup \{i\}$ to σ), this value is restored by `or-tools` giving access to the cover of the (restored) partial solution.

Rule 3. The inclusion between two covers $\mathcal{V}_{\mathcal{D}}^{\sigma^+}(i) \subseteq \mathcal{V}_{\mathcal{D}}^{\sigma^+}(j)$ is rewritten as $\mathcal{V}_{\mathcal{D}}^{\sigma^+}(i) \cap \mathcal{V}_{\mathcal{D}}^{\sigma^+}(j) = \mathcal{V}_{\mathcal{D}}^{\sigma^+}(i)$, and the intersection is performed by a bitwise-AND.

5 Running example

In this section, we illustrate the propagation of our CLOSEDPATTERN constraint and the difference that exists comparing to the use of a simple Reified Constraint Model (denoted by RCM, and detailed in Section 3). For that, let us take the transactional dataset given in Table 1: Five transactions t_1 to t_5 and eight items from *A* to *H*.

Figure 1 shows the tree search explored using CLOSEDPATTERN (**part(a)**) and the tree search explored using a reified model (**part(b)**) to extract closed

frequent patterns at minimum support $\theta = 2$. Here, both approaches use the same branching heuristics, namely `Lex` on variables and `Max_val` on values. First of all, it is worth noticing that the search space that can be explored using `CLOSEDPATTERN` is defined only on decision variables (item variables), whereas the reified model adds a further dimension with auxiliary variables (transaction variables).

At the root node (node 1), no pruning is done since all items are frequent and no item is a full extension of the empty pattern (see Table 1). Thereafter, `CLOSEDPATTERN` and `RCM` are acting in the same manner on the branch $A = 1$. With $A = 1$, B, C, E, F, G, H become infrequent. That is, the 1 values are pruned (*rule 2*). With `RCM` on node 2, the pruning on the five decision variables (the item variables) induce a pruning on four auxiliary variables (transaction variables). On the branch $A = 1$, two solutions are found: $\langle AD \rangle$ and $\langle A \rangle$.

Branching on $A = 0$ (node 5), the value 1 of D is pruned with *rule 3* of `CLOSEDPATTERN`. From Table 1, we have $D \Rightarrow A$, and a branching on $A = 0$ reduces D to 0. Here, we can say that the DC is maintained on node 5 using `CLOSEDPATTERN`, which is not the case using `RCM`. The same observation can be made on nodes 7, 9 and 11.

Let us take the node 6, here the branching on $A = 0$ and thereafter on $B = 1$ will make C, D, E, F, H infrequent (*rule 2*). Moreover, (*rule 1*) can be applied since G is a full extension of B (i.e., we cannot have a frequent closed pattern including B without G). That is, the value 0 is pruned from the domain of G , which allows us to reach the solution $\langle BG \rangle$. The same observation can be made on node 8.

To sum up, `FILTER-CLOSEDPATTERN` maintains DC at each node and thus, enumerates the solutions backtrack-free (no fails). The same cannot be said with `RCM` because the *rule 3* is never covered in this example and there are 3 fails.

6 Experiments

We made several experiments to compare and evaluate our global constraint with the state of the art methods (CP and specialized methods).

Benchmark datasets. We selected several real and synthetic datasets [17,4] from FIMI repository⁴ with large size. These datasets have varied characteristics representing different application domains. Table 2 reports for each dataset, the number of transactions $|\mathcal{T}|$, the number of items $|\mathcal{I}|$, the average size of transactions $|\widehat{\mathcal{T}}|$, its density ρ (i.e., $|\widehat{\mathcal{T}}|/|\mathcal{I}|$) and its *size* (i.e., $|\mathcal{T}| \times |\mathcal{I}|$). We note that the datasets are presented according to their size. They represent various numbers of transactions, numbers of items, densities. We have datasets that are very dense like Chess and Connect (resp. 49% and 33%), and others that are very sparse like Retail and BMS-Web-View1 (resp. 0.06% and 0.5%). Note that we have datasets of sizes going from $\approx 10^5$ to more than 10^9 .

⁴ <http://fimi.ua.ac.be/data/>

Table 2: Dataset Characteristics.

Dataset	$ \mathcal{T} $	$ \mathcal{I} $	$ \widehat{\mathcal{T}} $	ρ	type of data	size
Chess	3 196	75	37	49%	game steps	239 700
Splice1	3 190	287	60	21%	genetic sequences	915 530
Mushroom	8 124	119	23	19%	species of mushrooms	966 756
Connect	67 557	129	43	33%	game steps	8 714 853
BMS-Web-View1	59 602	497	2.5	0.5%	web click stream	29 622 194
T10I4D100K	100 000	1 000	10	1%	synthetic dataset	100 000 000
T40I10D100K	100 000	1 000	40	4%	synthetic dataset	100 000 000
Pumsb	49 046	7 117	74	1%	census data	349 060 382
Retail	88 162	16 470	10	0.06%	retail market basket data	1 452 028 140

Experimental protocol. The implementation of our approach was carried out in the `or-tools` solver. All experiments were conducted on an Intel Xeon E5-2680 @ 2.5 GHz with 128 Gb of RAM with a timeout of 3600s. For each dataset, we decreased the (relative) θ threshold until it is impossible to extract all closed patterns within the allocated time/memory. We have implemented two variants of CLOSEDPATTERN constraint: **(i)** CLOSEDPATTERN-DC ensuring DC with *rules 1, 2 and 3* (cubic pruning). **(ii)** CLOSEDPATTERN-WC ensuring a weaker consistency with only *rules 1 and 2* (quadratic pruning). Comparisons are made with: **(i)** CP4IM, the state-of-the-art on CP approaches, that uses an RCM model. **(ii)** LCM, the state-of-the-art on specialized methods.

For CLOSEDPATTERN and CP4IM, we use the same branching heuristics, namely `Lex` on variables and `Max_val` on values. We experimented using the available distributions of LCM-v5.3,⁵ and CP4IM,⁶ with `Gecode` as the underlying solver of CP4IM. Table 3 gives a comparison between CLOSEDPATTERN (WC and DC versions), CP4IM and LCM. We report the number of closed patterns $\#C$ of each instance, the number of propagations, the number of nodes, the CPU times in seconds and the number of failures.

CLOSEDPATTERN (DC vs WC). Despite the pruning complexity, DC clearly dominates WC in terms of CPU times (except for *BMS1* dataset where both are more or less equivalent). For instance, on the *pumsb* dataset with $\theta = 70\%$, DC is about 4 times faster than WC. As a second observation, the use of *rule 3* can reduce drastically the number of explored nodes and thus, number of propagations. For instance, we note a reduction of 38% on explored nodes on *connect* and 98% on *splice1* compared to WC.

CLOSEDPATTERN vs CP4IM. If we compare CLOSEDPATTERN with CP4IM, the main observation is that DC outperforms significantly CP4IM at all levels. In terms of CPU times and without counting the Out-of-memory instances, we can observe 23 instances (out of 30) with a speed-up factor between 2 and 15. Factors of 27 to 45 are noted for six instances and for one instance, we have a factor of 182. The weaker version (WC) is also better than CP4IM except two instances in *connect* and two instances in *chess*. In terms of number of propagations, we observe a gain factor within a range from 13 to 300. This is because of

⁵ <http://research.nii.ac.jp/~uno/codes.htm>

⁶ <https://dtai.cs.kuleuven.be/CP4IM/>

Table 3: CLOSEDPATTERN vs CP4IM vs LCM. (OOM: Out Of Memory; TO: TimeOut; (1): CLOSEDPATTERN-WC; (2): CLOSEDPATTERN-DC; (3): CP4IM)

\mathcal{D}	θ (%)	#C (\approx)	#Propagations			#Nodes			Time (s)			Failures			
			(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	LCM	(1)	(2)	(3)
chess	50	10^6	4 037 618	920 377	9 201 740	4 037 211	738 901	738 907	7.26	3.21	10.97	0.32	1 649 155	0	3
	40	10^6	15 931 714	3 411 690	30 541 475	15 929 751	2 733 667	2 733 735	26.37	12.27	40.85	1.44	6 598 042	0	34
	30	10^6	71 744 915	13 298 679	105 040 377	71 734 057	10 679 631	10 681 739	109.36	45.92	136.31	6.07	30 527 213	0	1 054
	20	10^6	305 446 222	56 550 872	385 399 747	305 374 413	45 837 171	45 901 933	480.52	187.89	467.52	27.55	1.29 $\times 10^8$	0	32 381
	10	10^8	1.51×10^9	304 335 522	1.66×10^9	1.51×10^9	247 960 091	249 411 325	2 287.98	969.40	1 950.51	141.55	6.34×10^8	0	725 617
splice1	20	10^2	54 055	25 078	913 779	53 991	487	487	0.10	0.59	22.59	0.04	26 752	0	0
	10	10^2	303 405	57 623	1 810 466	303 359	3 211	3 211	0.18	0.14	25.54	0.07	150 074	0	0
	5	10^4	5 013 077	1 706 269	36 211 067	5 013 031	63 935	63 935	3.39	3.23	138.54	1.46	2 474 548	0	0
	1	10^7	754 074 683	193 156 761	2.43×10^9	754 067 617	13 454 755	13 467 247	502.77	400.10	1 652.41	53.59	3.70×10^8	0	6 246
	mushroom	50	10^2	6 323	1 530	430 016	6 179	853	1 039	0.46	0.64	0.77	0.00	2 663	0
20		10^3	34 375	4 414	1 031 778	34 113	2 405	3 083	0.92	0.56	2.36	0.01	15 854	0	339
10		10^4	148 429	17 456	2 771 719	147 623	9 793	13 281	0.38	0.15	5.32	0.04	68 915	0	1 744
5		10^4	496 643	46 165	5 574 143	494 559	25 707	36 495	0.84	0.41	10.98	0.09	234 426	0	5 394
1		10^5	2 347 734	178 834	13 813 312	2 339 109	103 343	168 999	3.42	1.74	24.06	0.22	1 117 883	0	32 828
connect	0.5	10^5	3 896 158	264 660	18 230 838	3 883 697	156 613	265 781	5.00	3.62	29.84	0.34	18 635 42	0	54 584
	0.1	10^5	7 148 533	516 041	31 222 435	7 130 827	328 233	529 289	9.96	5.40	41.38	0.47	3 401 297	0	100 528
	0.05	10^5	8 001 925	613 712	36 520 438	7 983 119	407 763	622 145	9.94	6.37	43.69	0.51	3 787 678	0	107 191
	90	10^3	12 377	10 272	1 865 236	9 725	6 973	6 973	2.45	0.92	7.10	0.22	1 376	0	0
	80	10^4	60 312	48 156	9 455 886	44 771	30 223	30 223	9.35	1.65	16.57	0.31	7 274	0	0
BMSB	70	10^4	158 817	120 184	24 969 205	115 335	71 761	71 761	24.26	4.09	33.72	0.40	21 787	0	0
	60	10^5	297 175	240 634	51 648 114	202 637	136 699	136 699	45.23	7.30	45.73	0.39	32 969	0	0
	50	10^5	570 870	469 442	98 602 318	378 603	260 223	260 223	85.99	14.53	110.19	0.52	59 190	0	0
	40	10^5	1 139 466	886 722	177 653 293	762 579	478 781	478 781	159.04	27.32	153.39	0.83	141 899	0	0
	30	10^5	2 173 593	1 752 199	313 288 691	1 400 609	920 823	920 823	263.64	49.97	304.52	1.37	239 893	0	0
TUP	20	10^6	8 440 998	5 637 016	633 429 969	6 092 697	2 966 399	2 966 399	678.95	157.40	712.68	4.37	1 563 149	0	0
	10	10^7	55 152 559	31 370 763	1.76×10^9	41 641 659	16 075 555	16 075 555	3 110.90	760.71	2 597.89	17.70	12 783 052	0	0
	0.20	10^3	311 638	105 403	OOM	250 959	1 677	OOM	111.04	145.72	OOM	0.04	124 671	0	OOM
	0.16	10^3	493 069	116 902	OOM	432 419	2 617	OOM	161.60	218.88	OOM	0.06	214 939	0	OOM
	0.12	10^3	921 446	134 436	OOM	860 811	4 971	OOM	239.08	202.46	OOM	0.05	427 940	0	OOM
TAP	0.08	10^4	3 638 431	200 049	OOM	3 577 735	20 787	OOM	2 441.84	1 899.70	OOM	0.11	1 778 940	0	OOM
	1	10^1	1 000	1 000	OOM	1	1	OOM	0.46	0.24	OOM	0.03	0	0	OOM
	1	10^2	155 166	76 955	OOM	154 559	805	OOM	3.75	1.14	OOM	0.14	76 894	0	OOM
	0.5	10^3	509 852	174 015	OOM	509 441	2 185	OOM	10.29	3.11	OOM	0.42	253 647	0	OOM
	0.1	10^3	11 228 933	520 218	OOM	11 228 667	53 625	OOM	258.44	11.30	OOM	0.94	5 587 527	0	OOM
pumba	0.05	10^4	21 004 412	1 257 876	OOM	21 003 915	93 987	OOM	477.22	27.96	OOM	1.30	10 454 964	0	OOM
	0.01	10^5	130 686 806	17 313 642	OOM	130 673 093	566 795	OOM	1 471.32	905.17	OOM	3.49	65 053 149	0	OOM
	10	10^2	8 712	4 900	OOM	7 801	177	OOM	0.89	1.13	OOM	0.43	3 818	0	OOM
	5	10^2	97 982	47 983	OOM	97 289	643	OOM	2.34	1.78	OOM	1.31	48 328	0	OOM
	1	10^5	19 932 041	937 766	OOM	19 931 799	130 477	OOM	599.49	25.78	OOM	21.32	9 900 663	0	OOM
retail	0.5	10^6	TO	6 394 931	OOM	TO	2 551 883	OOM	TO	953.58	OOM	13.31	TO	0	OOM
	95	10^3	7 981	7 438	OOM	877	223	OOM	2.85	1.45	OOM	0.20	328	0	OOM
	90	10^3	17 429	11 074	OOM	10 323	2 933	OOM	44.92	13.37	OOM	0.29	3 695	0	OOM
	85	10^4	61 459	29 149	OOM	54 331	17 027	OOM	155.79	58.84	OOM	0.27	18 652	0	OOM
	80	10^4	176 955	89 805	OOM	169 743	66 615	OOM	640.59	133.97	OOM	0.33	51 564	0	OOM
retail	75	10^5	472 624	249 382	OOM	465 267	202 165	OOM	1 010.43	271	OOM	0.48	131 551	0	OOM
	70	10^5	969 374	567 929	OOM	961 763	482 517	OOM	2 150.80	509.79	OOM	0.69	239 623	0	OOM
	10	10	16 501	16 490	OOM	37	19	OOM	1	2.55	OOM	0.06	9	0	OOM
	1	10^2	31 089	19 852	OOM	14 695	329	OOM	11.03	4.02	OOM	0.10	7 188	0	OOM
	0.5	10^3	205 688	56 868	OOM	189 475	1 233	OOM	61.41	12.73	OOM	0.32	94 156	0	OOM
0.1	10^4	23 520 823	4 506 219	OOM	23 506 749	15 901	OOM	495.52	796.82	OOM	0.80	11 745 679	0	OOM	
0.05	10^4	114 357 067	18 342 468	OOM	114 345 005	40 229	OOM	2 352.14	2 645.06	OOM	1.07	57 152 804	0	OOM	

the huge number of propagator calls for reified constraints comparing to one propagator call using CLOSEDPATTERN. The number of explored nodes is also reduced, sometimes by half (e.g., *mushroom*), using CLOSEDPATTERN-DC. This is not a surprise as its number of explored nodes is optimal. Another observation is the experimental validation of Proposition 3: CLOSEDPATTERN-DC extracts the closed patterns in a *backtrack-free* manner. All solutions are enumerated without any fail (see failures column in Table 3). CP4IM requires an important number of backtracks on most datasets. On *connect* and three instances of *splice1* we can observe that CLOSEDPATTERN-DC and CP4IM explore the same number of nodes. The reified model on such dense datasets and using (Lex\Max_val) heuristics is able to prune all inconsistent values. This result is confirmed by the

number of failures always equal to zero (backtrack-free). Even if the WC version is faster, CP4IM remains better in terms of pruning (#nodes). Finally, the major drawback using the reified model for frequent pattern extraction is the memory consumption. We denote 25 Out-of-memory (out of 51 instances). The Out-of-memory state is due to the huge number of reified constraints. For instance, if we take the *T40I10D100K* dataset, the CP model produced by CP4IM contains $|\mathcal{T}| + |\mathcal{I}| = 101\,000$ variables, $|\mathcal{T}| = 100\,000$ reified constraints to express the channeling constraints, $2 \times |\mathcal{I}| = 2 \times 1\,000$ reified constraints to express the closure and frequency constraints. This means that the CP solver has to load in memory a CP model of 102 000 reified constraints expressed on 101 000 variables, which represents the *size* given in Table 2. From Table 2, we observe that **Gecode** is not able to handle CP4IM models on datasets of size greater than $\approx 10^7$ (greater than *connect* dataset).

CLOSEDPATTERN vs LCM. In terms of CPU times, LCM remains the leader on basic queries. However, CLOSEDPATTERN is quite competitive as a declarative approach. For instance, if we take *chess* dataset, LCM is 15 times faster than CLOSEDPATTERN-DC on average, where it is more than 120 times faster on average comparing to CP4IM. CLOSEDPATTERN pruning acts only within the current node of the search tree, without imposing any condition on the main search algorithm such as variable or value orderings. This allows to consider new constraints and let the main search algorithm adopt the best heuristics favouring the whole solving. To illustrate our point, we propose to model a particular problem (k -pattern sets) in a declarative manner where LCM could not meet this need.

k -patterns instance. A promising road to discover useful patterns is to impose constraints on a set of k related patterns (k -pattern sets) [5,9]. In this setting, the interest of a pattern is evaluated w.r.t. a set of patterns. We propose to model and solve a particular instance, coined `dist_kpatterns_lb_ub`. Here, we aim at finding k closed patterns $\{P^1, \dots, P^k\}$ s.t:

- (i) $\forall i \in [1, k] : \text{CLOSEDPATTERN}(P^i)$ (Closed Frequent Patterns),
- (ii) $\forall i, j \in [1, k] : P^i \cap P^j = \emptyset$ (all distinct patterns constraints),
- (iii) $\forall i \in [1, k] : \text{lb} < |P^i| < \text{ub}$ (min and max size constraints).

Figure 2 shows a comparison between two models, M1 using CLOSEDPATTERN for the CFPM part of the problem, M2 using a CP4IM implementation. We selected two dataset instances where CP4IM does not reach the Out-of-memory state and where we have a reasonable number of closed patterns, *chess* with $\theta = 80\%$ (5084 closed patterns) and *connect* with $\theta = 90\%$ (3487 closed patterns), and we have varied k with a timeout of 3600s. After a few preliminary tests, the bounds `lb` and `ub` on the size of the patterns were set to 2 and 10 respectively. On *chess*, model M1 is robust and scales well: it is linear on k and never exceeds 6 *min* even with $k = 12$ (323.53s). M2 follows an exponential scale and goes beyond the timeout with only $k = 8$ (7222.41s). The same observation can be made on the **connect** instance, but in a more pronounced way on the exponential scale followed by M2. With $k = 4$, M2 goes beyond the timeout with 5428.05s whereas M1 confirms its linear behavior when varying k from 2 to 12.

For such problems, a baseline can be the use of specialized methods with postprocessing. One can imagine (i) the use of LCM to extract the total number of closed patterns and (ii) a generate-and-test search trying to find distinct patterns of a given size. Such approach can be very expensive. Here, the postprocessing will generate all the possible k combinations of closed patterns. For instance, we recall that for *chess* with $\theta = 80\%$ we have 5084 closed patterns. With $k = 12$ and using M1, we need less than 6 min, where using the baseline we have to cope with a massive number of combinations. Thus, this last experiment confirms that if LCM is faster on basic queries (e.g., asking for closed frequent with given size), it cannot cope with complex queries. It would need to think and to propose an adhoc solution whereas CP enables a novice DM-user to express his query as constraints.

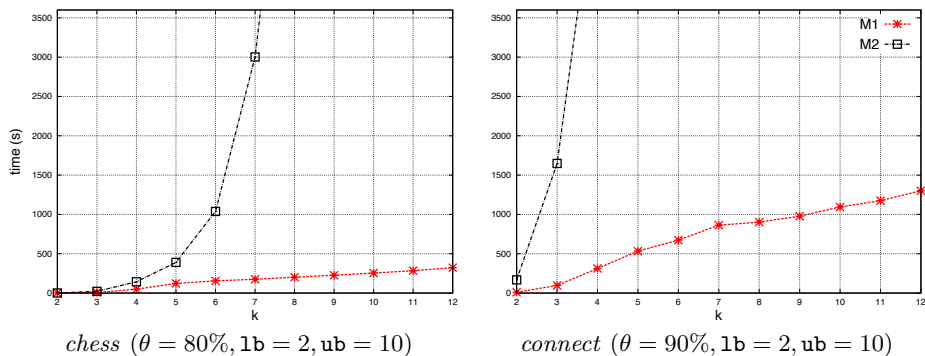


Fig. 2: `dist_kpatterns_1b_ub` instance using `CLOSEDPATTERN` and `CP4IM`.

7 Conclusion

In this paper we have introduced a new global constraint for Closed Frequent Pattern Mining. The `CLOSEDPATTERN` constraint captures the particular semantic of the CFPM problem, namely the minimum frequency and closedness of patterns. To propagate efficiently this global constraint, we have first defined three filtering rules that ensure domain consistency. Second, we have defined a filtering algorithm that establishes domain consistency in a *cubic* time complexity and *quadratic* space complexity. We have implemented this filtering algorithm into the `or-tools` solver using a vertical representation of datasets and smart data structures. We have conducted an experimental study on several real and synthetic datasets, showing the efficiency and the scalability of the global constraint compared to a reified constraints approach such as `CP4IM`. Finally, to show the applicability and the flexibility of `CLOSEDPATTERN` compared to specialized methods we performed experiments on an instance of k -pattern set problem where `CLOSEDPATTERN` is integrated with a set of constraints.

References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993. pp. 207–216. ACM Press (1993)
2. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: Generalizing association rules to correlations. In: SIGMOD. pp. 265–276 (1997)
3. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for itemset mining. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 204–212. ACM (2008)
4. Grahne, G., Zhu, J.: Fast algorithms for frequent itemset mining using FP-Trees. *IEEE Trans. Knowl. Data Eng.* 17(10), 1347–1362 (2005)
5. Guns, T., Nijssen, S., De Raedt, L.: k-pattern set mining under constraints. *Knowledge and Data Engineering, IEEE Transactions on* 25(2), 402–418 (2013)
6. Guns, T., Nijssen, S., De Raedt, L.: Itemset mining: A constraint programming perspective. *Artificial Intelligence* 175(12), 1951–1983 (2011)
7. Hoeve, W., Katriel, I.: Global constraints. In: *Handbook of Constraint Programming*, pp. 169–208. Elsevier Science Inc. (2006)
8. Kemmar, A., Loudni, S., Lebbah, Y., Boizumault, P., Charnois, T.: PREFIX-PROJECTION global constraint for sequential pattern mining. In: CP 2015. LNCS, vol. 9255, pp. 226–243. Springer (2015)
9. Khiari, M., Boizumault, P., Crémilleux, B.: Constraint programming for mining n-ary patterns. In: CP 2010. pp. 552–567 (2010)
10. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery* 1(3), 241–258 (1997)
11. Nijssen, S., Guns, T.: Integrating constraint programming and itemset mining. In: *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II. Lecture Notes in Computer Science*, vol. 6322, pp. 467–482. Springer (2010)
12. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. *Inf. Syst.* 24(1), 25–46 (1999)
13. Pei, J., Han, J., Mao, R.: CLOSET: an efficient algorithm for mining frequent closed itemsets. In: *SIGMOD Workshop on Data Mining and Knowledge Discovery*. pp. 21–30 (2000)
14. Uno, T., Asai, T., Uchida, Y., Arimura, H.: An efficient algorithm for enumerating closed patterns in transaction databases. In: DS 2004. pp. 16–31 (2004)
15. Wang, J., Han, J., Pei, J.: CLOSET+: searching for the best strategies for mining frequent closed itemsets. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003*. pp. 236–245 (2003)
16. Zaki, M.J., Gouda, K.: Fast vertical mining using diffsets. In: *SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 326–335 (2003)
17. Zaki, M.J., Hsiao, C.: CHARM: an efficient algorithm for closed itemset mining. In: *SIAM International Conference on Data Mining*. pp. 457–473 (2002)