A Relaxation-based Approach for Mining Diverse Closed Patterns

Arnold Hien[2], Samir Loudni[2,3]✉, Noureddine Aribi[1], Yahia Lebbah[1], Mohammed Laghzaoui[1], Abdelkader Ouali[2], and Albrecht Zimmermann[2]

[1] University of Oran1, Lab. LITIO, 31000 Oran, Algeria
[2] Normandie Univ., UNICAEN, CNRS – UMR GREYC, France
[3] TASC (LS2N-CNRS), IMT Atlantique, FR – 44307 Nantes, France

Abstract. In recent years, pattern mining has moved from a slow-moving repeated three-step process to a much more agile iterative/user-centric mining model. A vital ingredient of this framework is the ability to *quickly* present a set of *diverse* patterns to the user. In this paper, we use constraint programming (well-suited to user-centric mining due to its rich constraint language) to efficiently mine a diverse set of closed patterns. Diversity is controlled through a threshold on the Jaccard similarity of pattern occurrences. We show that the Jaccard measure has no monotonicity property, which prevents usual pruning techniques and makes classical pattern mining unworkable. This is why we propose anti-monotonic lower and upper bound relaxations, which allow effective pruning, with an efficient branching rule, boosting the whole search process. We show experimentally that our approach significantly reduces the number of patterns and is very efficient in terms of running times, particularly on dense data sets.

1 Introduction

The original data analysis model based on pattern mining consists of three steps in a kind of *multi-waterfall cycle*: 1) a user chooses the values of one or several mining parameters, 2) an underlying engine extracts patterns (often taking not inconsiderable time to do so), and 3) the user sifts through a (potentially very large) set of result patterns and interprets them, using their insights to return to the first step and repeat the cycle.

Recently, this approach has been challenged by an increasing focus on *user-centered*, *interactive*, and *anytime* pattern mining [14]. This new paradigm stresses that users should be presented quickly with patterns likely to be interesting to them, and typically affect later iterations of the mining process by giving feedback. A powerful framework for taking a variety of user feedback into account is pattern mining via constraint programming (CP). Much of the current focus in this domain is on user-centered/interactive mining, particularly the ability to elicit and exploit user feedback [9, 14, 18]. An important aspect of requesting such feedback is that the user be quickly presented with *diverse* results. If patterns are too similar to each other, deciding which one to prefer can become challenging, and if they appear in several successive iterations, it eventually becomes a slog. Similarly, a method that produces diverse results but takes a long time to do so, risks that the user checks out of the process. Older work on diversity either post-process patterns derived from the process described above [5,12,21], use heuristics [20] or view

it purely from the point of view of speeding up the extraction process [8]. Recent work, on the other hand, pushes diversity constraints into the mining process itself [3, 4]. At the algorithmic level, additional user-specified constraints often require new implementations to filter out the patterns violating or satisfying the user's constraints, which can be computationally infeasible for large databases.

In the last decade, data mining has been combined with constraint programming to model various data mining problems [2,6,13,19]. The main advantage of CP for pattern mining is its declarativity and flexibility, which include the ability to incorporate new user-specified constraints without the need to modify the underlying system. Moreover, CP allows to define flexible search strategies.[4] In this paper, we propose to add to the literature on *explicitly* taking the diversity of patterns (in terms of the data instances they describe) into account and to use an exhaustive process to find candidates for inclusion into a result set. To achieve this, we use the widely accepted Jaccard index to compare patterns and formulate a diversity constraint, which has no monotonicity property, implying limited pruning during search. To cope with this problem, we propose two anti-monotonic relaxations: (i) A lower bound relaxation, which allows to prune non-diverse items during search. This is integrated in our constraint programming based approach through a new global constraint taking into account diversity with its filtering algorithms (aka, propagators); (ii) An upper bound relaxation to find items ensuring diversity. This is exploited through a new branching rule, boosting the search process towards diverse patterns. We demonstrate the performance of our proposed method experimentally, comparing to the state-of-the-art in CP-based closed pattern mining.

2 Preliminaries

2.1 Itemset Mining

Let $\mathcal{I} = \{1, ..., n\}$ be a set of n *items*, an *itemset* (or pattern) P is a non-empty subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L}_\mathcal{I} = 2^\mathcal{I} \backslash \emptyset$. A transactional dataset $\mathcal{D}$ is a bag (or multiset) of transactions over $\mathcal{I}$, where each *transaction* t is a subset of $\mathcal{I}$, i.e., $t \subseteq \mathcal{I}$; $\mathcal{T} = \{1, ..., m\}$ a set of m *transaction* indices. An itemset P *occurs* in a transaction t, iff $P \subseteq t$. The *cover* of P in $\mathcal{D}$ is the set of transactions in which it occurs: $\mathcal{V}_\mathcal{D}(P) = \{t \in \mathcal{D} \mid p \subseteq t\}$. The *support* of P in $\mathcal{D}$ is the size of its cover: $sup_\mathcal{D}(P) = |\mathcal{V}_\mathcal{D}(P)|$. An itemset P is said to be *frequent* when its support exceeds a user-specified minimal threshold θ, $sup_\mathcal{D}(P) \geq \theta$. Given $S \subseteq \mathcal{D}$, $items(S)$ is the set of common items belonging to all transactions in S: $items(S) = \{i \in \mathcal{I} \mid \forall t \in S, i \in t\}$. The *closure* of an itemset P, denoted by $Clos(P)$, is the set of common items that belong to all transactions in $\mathcal{V}_\mathcal{D}(P)$: $Clos(P) = \{i \in \mathcal{I} \mid \forall t \in \mathcal{V}_\mathcal{D}(P), i \in t\}$. An itemset P is said to be *closed* iff $Clos(P) = P$. Constraint-based pattern mining aims at extracting all patterns P of $\mathcal{L}_\mathcal{I}$ satisfying a selection predicate c (called *constraint*) which is usually called *theory* [5]: $Th(c)$. A common example is the frequency measure leading to the minimal support constraint, which can be combined with the closure constraint to mine closed frequent itemsets.

[4] Opposed to more rigid search in classical pattern mining algorithms, which often rely on exploiting the properties of a particular constraint.

Example 1. Figure 1 shows the itemset lattice derived from a toy dataset with five items and 100 transactions. As the figure shows, there exist 26 frequent closed itemsets with $\theta = 7$.

Most constraint-based mining algorithms take advantage of monotonicity which offers pruning conditions to safely discard non-promising patterns from the search space. Several frameworks exploit this principle to mine with a monotone or an anti-monotone constraint. Other classes of constraints have also been considered [15, 16]. However, for constraints that are not anti-monotone, pushing them into the discovery algorithm might lead to less effective pruning phases. Thus, we propose in this paper to exploit the *witness* concept introduced in [11] to handle such constraints. A witness is a single itemset on which we can test whether a constraint holds and derive information about properties of other itemsets.

Definition 1 (Witness). *Let P, Q itemsets, and $C : \mathcal{I} \mapsto \{true, false\}$, then W, $P \subseteq W \subseteq P \cup Q$, is called a positive (negative) witness iff $\forall P', P \subseteq P' \subseteq P \cup Q : C(W) = true \Rightarrow C(P') = true$ ($C(W) = false \Rightarrow C(P') = false$).*

2.2 Diversity of Itemsets

The Jaccard index is a classical similarity measure on sets. We use it to quantify the overlap of the covers of itemsets.

Definition 2 (Jaccard index). *Given two itemsets P and Q, the Jaccard index is the relative size of the overlap of their covers : $Jac(P, Q) = \frac{|\mathcal{V}_{\mathcal{D}}(P) \cap \mathcal{V}_{\mathcal{D}}(Q)|}{|\mathcal{V}_{\mathcal{D}}(P) \cup \mathcal{V}_{\mathcal{D}}(Q)|}$.*

A lower Jaccard indicates low similarity between itemset covers, and can thus be used as a measure of diversity between pairs of itemsets.

Definition 3 (Diversity/Jaccard constraint). *Let P and Q be two itemsets. Given the Jac measure and a diversity threshold J_{max}, we say that P and Q are pairwise diverse iff $Jac(P, Q) \leq J_{max}$. We will denote this constraint by c_{Jac}.*

Our aim is to push the Jaccard constraint during pattern discovery to prune non-diverse itemsets. To achieve this, we maintain a *history* $\mathcal{H}$ of extracted pairwise diverse itemsets during search and constrain the next mined itemsets to respect a maximum Jaccard constraint with all itemsets already included in $\mathcal{H}$. This problem can be formalized as follows.

Definition 4 (k diverse frequent itemsets). *Given a current history $\mathcal{H} = \{H_1, \ldots, H_k\}$ of k pairwise diverse frequent closed itemsets, the Jac measure and a diversity threshold J_{max}, the task is to mine new itemsets P such that $\forall\, H \in \mathcal{H}$, $Jac(P, H) \leq J_{max}$.*

Example 2. The lattice in Figure 1 depicts the set of diverse FCIs (marked with blue and green solid line circles) with $J_{max} = 0.19$ and $\mathcal{H} = \{BE\}$. ACE is a diverse FCI (i.e., $Jac(ACE, BE) = 0.147 < 0.19$).

Proposition 1. *Let P, Q and P' be three itemsets s.t. $P \subset P'$. $Jac(P, Q)$ may be smaller, equal or greater than $Jac(P', Q)$.*

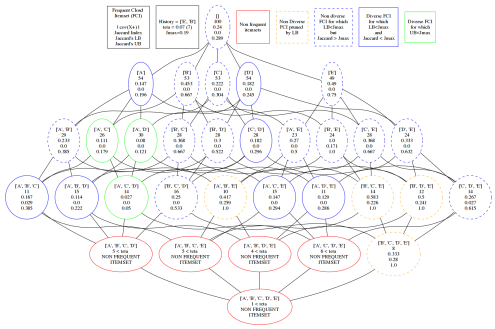


Fig. 1: The powerset lattice of frequent closed itemsets ($\theta = 7$) for the dataset $\mathcal{D}$ of Example 1.

Based on the above proposition, the anti-monotonicity of the maximum Jaccard constraint does not hold, which disables pruning. Thus, instead of solving the problem of Definition 4 directly, we introduce bounds in Section 3 that allow us to prune the search space using a relaxation of the Jaccard constraint. The appeal of this approach is that we are able to infer monotone and anti-monotone properties from this relaxation.

2.3 Constraint Programming (CP)

Constraint programming [10] is a powerful paradigm which offers a generic and modular approach to model and solve combinatorial problems. A CP model consists of a set of variables $X = \{x_1, \ldots, x_n\}$, a set of domains D mapping each variable $x_i \in X$ to a finite set of possible values $dom(x_i)$, 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 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 constraints. Constraint solvers typically use backtracking search to explore the search space of partial assignments. Algorithm 1 provides a general overview of a CP solver. At each node of the search tree, procedure *Constraint-Search* selects an unassigned variable (line 8) according to user-defined heuristics and assigns it a value (line 9). It backtracks when a constraint cannot be satisfied, i.e. when at least one domain is empty (line 5). A solution is obtained (line 12) when each domain $dom(x_i)$ is reduced to a singleton and all constraints are satisfied. The main concept used to speed up the search is constraint propagation by *Filtering algorithms*. At each assignment, constraint 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 $v \in dom(x_i)$, there is an assignment satisfying c such that $(x_i = v)$. Global constraints are families of constraints defined by a relation on any number of variables [10].

2.4 A CP Model for Frequent Closed Itemset Mining

The first constraint programming model for frequent closed itemset mining (FCIM) was introduced in [6]. It is based on reified constraints to connect item variables to transac-

Algorithm 1: Constraint-Search(D)

```
1  In: X : a set of decision variables; C : a set of constraints;
2  InOut: D : a set of variable domains;
3  begin
4      D ← Filtering(D, C)
5      if there exists x_i ∈ X s.t. dom(x_i) is empty then
6          return failure
7      if there exists x_i ∈ X s.t. |dom(x_i)| > 1 then
8          Select x_i ∈ X s.t. |dom(x_i)| > 1
9          forall v ∈ dom(x_i) do
10             Constraint-Search(Dom ∪ {x_i → {v}})
11     else
12         output solution D
```

tion variables. The first global constraint CLOSEDPATTERNS for mining frequent closed itemsets was proposed in [13]. The global constraint COVERSIZE for computing the exact size of the cover of an itemset was introduced in [19]. It offers more flexibility in modeling problems. We present the global constraint CLOSEDPATTERNS.

Global Constraint CLOSEDPATTERNS. Most declarative methods use a vector x of Boolean variables $(x_1, \ldots, x_{|\mathcal{I}|})$ for representing itemsets, where x_i represents the presence of the item $i \in \mathcal{I}$ in the itemset. We will use the following notations: $x^+ = \{i \in \mathcal{I} \,|\, dom(x_i) = \{1\}\}$, $x^- = \{i \in \mathcal{I} \,|\, dom(x_i) = \{0\}\}$ and $x^* = \{i \in \mathcal{I} \,|\, i \notin x^+ \cup x^-\}$.

Definition 5 (CLOSEDPATTERNS). *Let x be a vector of Boolean variables, θ a support threshold and $\mathcal{D}$ a dataset. The global constraint* CLOSEDPATTERNS$_{\mathcal{D},\theta}(x)$ *holds if and only if x^+ is a closed frequent itemset w.r.t. the threshold θ.*

Definition 6 (Closure extension [22]). *A non-empty itemset P is a closure extension of Q iff $\mathcal{V}_{\mathcal{D}}(P \cup Q) = \mathcal{V}_{\mathcal{D}}(Q)$.*

Filtering of CLOSEDPATTERNS. [13] also introduced a complete filtering algorithm for CLOSEDPATTERNS based on three rules. The first rule filters 0 from $dom(x_i)$ if $\{i\}$ is a closure extension of x^+ (see Definition 6). The second rule filters 1 from $dom(x_i)$ if the itemset $x^+ \cup \{i\}$ is infrequent w.r.t. θ. Finally, the third rule filters 1 from $dom(x_i)$ if $\mathcal{V}_{\mathcal{D}}(x^+ \cup \{i\})$ is a subset of $\mathcal{V}_{\mathcal{D}}(x^+ \cup \{j\})$ where j is an absent item, i.e. $j \in x^-$.

To show the strength and the flexibility of the CP approach in taking into account user's constraints, we formulate a CP model to extract more specific patterns using the following four global constraints : $\mathcal{C} = \{$CLOSEDP$_{\mathcal{D},\theta}(X)$, ATLEAST(X, lb), KNAPSACK(X, z, w), REGULAR$(X, DFA)\}$ The ATLEAST constraint enforces that at least lb variables in X are assigned to 1; the KNAPSACK constraint restricts a weighted linear sum to be no more than a given capacity z, i.e. $\sum_i w_i X_i \leq z$; the REGULAR constraint imposes that X is accepted by deterministic finite automaton (DFA), which recognizes a regular expression.

Example 3. Let us consider the example of Figure 1 using $lb = 2$, $w = \langle 8, 7, 5, 14, 16 \rangle$, $z = 20$, $\theta = 7$ and the regular expression $0^*\,1^+\,0^*$ ensuring items' contiguity. Solving this CP model provides the solution: $Th(\mathcal{C}) = \{\{AB\}, \{BC\}, \{CD\}, \{ABC\}\}$.

3 A CP Model for Mining Diverse Frequent Closed Itemsets

We present our approach for computing diverse FCIs. The key idea is to compute an approximation of the set of diverse FCIs by defining two bounds on the Jaccard index that allow us to reduce the search space. All the proofs are given in the Supp. material [1].

3.1 Problem Reformulation

Proposition 1 states that the Jaccard constraint is neither monotonic nor anti-monotonic. So, we propose to approximate the theory of the original constraint c_{Jac} by a larger collection corresponding to the solution space of its *relaxation* c^r_{Jac}: $Th(c_{Jac}) \subseteq Th(c^r_{Jac})$. The key idea is to formulate a relaxed constraint having suitable monotonicity properties in order to exploit them for search space reduction. More precisely, we want to exploit *upper and lower bounding* operators to derive a monotone relaxation and an anti-monotone one of c_{Jac}.

Definition 7 (Problem reformulation). *Given a current history $\mathcal{H} = \{H_1, \ldots, H_k\}$ of extracted k pairwise diverse frequent closed itemsets, a diversity threshold J_{max}, a lower bound LB_J and an upper bound UB_J on the Jaccard index, the relaxed problem consists of mining candidate itemsets P such that $\forall H \in \mathcal{H}$, $LB_J(P, H) \leq J_{max}$. When $UB_J(P, H) \leq J_{max}$, for all $H \in \mathcal{H}$, the Jaccard constraint is fully satisfied.*

3.2 Jaccard Lower Bound

Let us now formalize how to compute the lower bound and how to exploit it. To arrive at a lower bound for the Jaccard value between two itemsets, we need to consider the situation where the *overlap* between them has become as small as possible, while the coverage that is proper to each itemset remains as large as possible.

Definition 8 (Proper cover). *Let P and Q be two itemsets. The proper cover of P w.r.t. Q is defined as $\mathcal{V}^{pr}_Q(P) = \mathcal{V}_\mathcal{D}(P) \backslash \{\mathcal{V}_\mathcal{D}(P) \cap \mathcal{V}_\mathcal{D}(Q)\}$.*

The lowest possible Jaccard would reduce the numerator to 0, which is however not possible under the minimum support threshold θ. The denominator, on the other hand, consists of $|\mathcal{V}_\mathcal{D}(H)|$ (which cannot change) and the part of P's coverage that does not overlap with H, i.e. $\mathcal{V}^{pr}_H(P)$.

Proposition 2 (Lower bound). *Consider a member pattern H of the history $\mathcal{H}$. Let P an itemset encountered during search such that $sup_\mathcal{D}(P) \geq \theta$, and $\mathcal{V}^{pr}_H(P)$ be the proper cover of P w.r.t. H. $LB_J(P, H) = \dfrac{\theta - |\mathcal{V}^{pr}_H(P)|}{|\mathcal{V}_\mathcal{D}(P)| + |\mathcal{V}_\mathcal{D}(H)| + |\mathcal{V}^{pr}_H(P)| - \theta}$ is a lower bound of $Jac(P, H)$.*

The lower bound on the Jaccard index enables us to discard some non-diverse itemsets, i.e., those with an LB_J value greater than J_{max} are *negative witnesses*.

Example 4. The set of all non diverse FCIs with a lower bound value greater than J_{max} are marked in Figure 1 with orange line circles.

Proposition 3 (Monotonicity of LB_J). *Let $H \in \mathcal{H}$ be an itemset. For any two itemsets $P \subseteq Q$, the relationship $LB_J(P,H) \leq LB_J(Q,H)$ holds.*

Property 3 establishes an important result to define a pruning condition based on the monotonicity of the lower bound (cf. Section 3.4). If $LB_J(P,H) > J_{max}$, then no itemset $Q \supseteq P$ will satisfy the Jaccard constraint (because LB_J is a lower bound), rendering the constraint itself *anti-monotone*. So, we can safely prune Q.

3.3 Jaccard Upper Bound

As our relaxation approximates the theory of the Jaccard constraint, i.e. $Th(c_{Jac}) \subseteq Th(c^r_{Jac})$, one could have itemsets P such that $LB_J(P,H) < J_{max}$ but $Jac(P,H) > J_{max}$ (see itemsets marked with blue dashed line circles in Figure 1). To tackle this case, we define an upper bound on the Jaccard index to evaluate the satisfaction of the Jaccard constraint, i.e., those with $UB_J(P,H) \leq J_{max}, \forall\, H \in \mathcal{H}$, are *positive witnesses*.

To derive the upper bound, we need to follow the opposite argument as for the lower bound: the highest possible Jaccard will be achieved if $\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)$ stays unchanged but the set $\mathcal{V}^{pr}_H(P)$ is reduced as much possible (under the minimum support constraint). If the intersection is greater than or equal θ, in the worst case scenario (leading to the highest Jaccard), a future P' covers only transactions in the intersection. If not, the denominator needs to contain a few elements of $\mathcal{V}^{pr}_H(P)$, $\theta - |\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)|$, to be exact.

Proposition 4 (Upper bound). *Given a member pattern H of the history $\mathcal{H}$, and an itemset P such that $sup_{\mathcal{D}}(P) \geq \theta$. $UB_J(P,H) = \dfrac{|\mathcal{V}_{\mathcal{D}}(H) \cap \mathcal{V}_{\mathcal{D}}(P)|}{|\mathcal{V}^{pr}_P(H)| + \max\{\theta, |\mathcal{V}_{\mathcal{D}}(H)| \cap |\mathcal{V}_{\mathcal{D}}(P)|\}}$ is an upper bound of $Jac(P,H)$.*

Example 5. The set of all diverse FCIs with UB_J values less than J_{max} are marked with green line circles in Figure 1.

Our upper bound can be exploited to evaluate the Jaccard constraint during mining. More precisely, in the enumeration procedure, if the upper bound of the current candidate itemset P is less than J_{max}, then c_{Jac} is fully satisfied. Moreover, if the upper bound is *monotonically decreasing* (or anti-monotonic), then all itemsets Q derived from P are also diverse (see Proposition 5).

Proposition 5 (Anti-monotonicity of UB_J). *Let H be a member pattern of the history $\mathcal{H}$. For any two itemsets $P \subseteq Q$, the relationship $UB_J(P,H) \geq UB_J(Q,H)$ holds.*

3.4 The Global Constraint CLOSEDDIVERSITY

This section presents our new global constraint CLOSEDDIVERSITY that exploits the LB relaxation to mine pairwise diverse frequent closed itemsets.

Algorithm 2: Filtering for CLOSEDDIVERSITY

```
1  In: θ, J_max : frequency and diversity thresholds; H : history of solutions encountered during search;
2  InOut: x = {x_1 ... x_n} : Boolean item variables;
3  begin
4      if ( |V_D(x^+)| < θ ∨ !PGrowth_LB(x^+, H, J_max)) then return false;
5      foreach i ∈ x^* do
6          if (|V_D(x^+ ∪ {i})| < θ) then
7              dom(x_i) ← dom(x_i) − {1}; x^-_Freq ← x^-_Freq ∪ {i}; x^* ← x^* \ {i}; continue;
8          if (|V_D(x^+ ∪ {i})| = |V_D(x^+)|) then
9              dom(x_i) ← dom(x_i) − {0}; x^+ ← x^+ ∪ {i};    x^* ← x^* \ {i};
10         if (! PGrowth_LB(x^+ ∪ {i}, H, J_max)) then
11             dom(x_i) ← dom(x_i) − {1}; x^-_Div ← x^-_Div ∪ {i}; x^* ← x^* \ {i}; continue;
12         foreach k ∈ (x^-_Freq ∪ x^-_Div) do
13             if (V_D(x^+ ∪ {i}) ⊆ V_D(x^+ ∪ {k})) then
14                 dom(x_i) ← dom(x_i) − {1}
15                 if k ∈ x^-_Freq then x^-_Freq ← x^-_Freq ∪ {i};
16                 else x^-_Div ← x^-_Div ∪ {i};
17                 x^* ← x^* \ {i}; break;
18     return true;
19 Function PGrowth_LB(x, H, J_max): Boolean
20     foreach H ∈ H do
21         if (LB_J(x, H) > J_max) then return false
22     return true
```

Definition 9 (CLOSEDDIVERSITY). *Let x be a vector of Boolean item variables, $\mathcal{H}$ a history of pairwise diverse frequent closed itemsets (initially empty), θ a support threshold, J_{max} a diversity threshold and $\mathcal{D}$ a dataset. The* CLOSEDDIVERSITY$_{\mathcal{D},\theta}(x, \mathcal{H}, J_{max})$ *global constraint holds if and only if: (1) x^+ is closed; (2) x^+ is frequent, $sup_{\mathcal{D}}(x^+) \geq \theta$; (3) x^+ is diverse, $\forall\, H \in \mathcal{H}, LB_J(x^+, H) \leq J_{max}$.*

Initially, the history $\mathcal{H}$ is empty. Our global constraint allows to incrementally update $\mathcal{H}$ with diverse FCIs encountered during search. Condition (3) expresses a necessary condition ensuring that x^+ is diverse. Indeed, one could have $LB_J(x^+, H) \leq J_{max}$ but $Jac(x^+, H) > J_{max}$. Thus, we propose in Section 4 to exploit our UB relaxation to guarantee the satisfaction of the Jaccard constraint.

The propagator for CLOSEDDIVERSITY exploits the filtering rules of CLOSEDPATTERNS (see Section 2.4). It also uses our LB relaxation to remove items i that cannot belong to a solution containing x^+. We denote by x^-_{Freq} the set of items filtered by the rule of infrequent items and by x^-_{Div} the set of items filtered by our LB rule.

Proposition 6 (CLOSEDDIVERSITY Filtering rule). *Given a history $\mathcal{H}$ of pairwise diverse frequent closed itemsets, a partial assignment on x, and a free item $i \in x^*$, $x^+ \cup \{i\}$ cannot lead to a diverse itemset if one of the two cases holds:*
1) if $\exists\, H \in \mathcal{H}$ s.t. $LB_J(x^+ \cup \{i\}, H) > J_{max}$, then we remove 1 from $dom(x_i)$.
2) if $\exists\, k \in x^-_{Div}$ s.t. $\mathcal{V}_{\mathcal{D}}(x^+ \cup \{i\}) \subseteq \mathcal{V}_{\mathcal{D}}(x^+ \cup \{k\})$, then $LB_J(x^+ \cup \{i\}, H) > LB_J(x^+ \cup \{k\}, H) > J_{max}$, thus we remove 1 from $dom(x_i)$.

Algorithm. The propagator for CLOSEDDIVERSITY is presented in Algorithm 2. It takes as input the variables x, the support threshold θ, the diversity threshold J_{max} and the current history $\mathcal{H}$ of pairwise diverse frequent closed itemsets. It starts by computing the cover of the itemset x^+ and checks if x^+ is either infrequent or not diverse (see function $\mathcal{P}Growth_{LB}$), if so the constraint is violated and a fail is returned (line 4). Algorithm 2 extends the filtering rules of CLOSEDPATTERNS (see Section 2.4) by examining the diversity condition of the itemset $x^+ \cup \{i\}$ (see Proposition 3). For each element $H \in \mathcal{H}$, the function $\mathcal{P}Growth_{LB}(x^+ \cup \{i\}, \mathcal{H}, J_{max})$ computes the value of $LB_J(x^+ \cup \{i\}, H)$ and tests if there exists an H s.t. $LB_J(x^+ \cup \{i\}, H) > J_{max}$ (lines 20-21). If so, we return $false$ (line 21) because $x^+ \cup \{i\}$ cannot lead to a diverse itemset w.r.t. $\mathcal{H}$, remove 1 from $dom(x_i)$ (line 11), update x^-_{Div} and x^* and we continue with the next free item. Otherwise, we return true. Second, we remove 1 from each free item variable $i \in x^*$ such that its cover is a superset of the cover of an absent item $k \in (x^-_{Freq} \cup x^-_{Div})$ (lines 12-17). The LB filtering rule associated to the case $k \in x^-_{Div}$ is a new rule taking its originality from the reasoning made on absent items.

Proposition 7 (Consistency and time complexity). *Algorithm 2 enforces Generalized Arc Consistency (GAC) (a.k.a. domain consistency [10]) in* $\mathcal{O}(n^2 \times m)$.

4 Using witnesses and the estimated frequency within the search

In this section, we show how to exploit the witness property and the estimated frequency so as to design a more informed search algorithm.

Positive Witness. During search, we compute incrementally the $UB(x^+ \cup \{i\}, H)$ of any extension of the partial assignment x^+ with a free item i. If, for each $H \in \mathcal{H}$, this upper bound is less or equal to J_{max}, then c_{Jac} is fully satisfied and $x^+ \cup \{i\}$ is a *positive witness*. Moreover, thanks to the anti-monotonicity of UB_J (see Proposition 5), all supersets of $x^+ \cup \{i\}$ will satisfy the Jaccard constraint.

Estimated frequency. The frequency of an itemset can be computed as the cardinality of the intersection of its items' cover: $sup_{\mathcal{D}}(x^+) = |\cap_{i \in x^+} \mathcal{V}_{\mathcal{D}}(i)|$, the intersection between 2 covers being performed by a bitwise-AND. To limit the number of intersections, we use an estimation of the frequency of each item $i \in \mathcal{I}$ w.r.t the set of present items x^+, denoted $eSup_{\mathcal{D}}(i, x^+)$. This estimation constitutes a *lower bound* of $|\mathcal{V}_{\mathcal{D}}(x^+ \cup \{i\})|$. Interestingly, if $eSup_{\mathcal{D}}(i, x^+) \geq \theta$ then $|\mathcal{V}_{\mathcal{D}}(x^+ \cup \{i\})| \geq \theta$, meaning that the intersection between covers is performed only if $eSup_{\mathcal{D}}(i, x^+) < \theta$, thereby leading to performance enhancement. In addition, we argue that the estimated support is an interesting heuristic to reinforce the witness branching rule. Indeed, branching on the variable having the minimum estimated support (using the lower bound of the real support) will probably activate our filtering rules (see Algorithm 2), thus reducing the search space. It will be denoted as MINCOV variable ordering heuristic.

We propose Algorithm 3 as a branching procedure (returns the next variable to branch on). When the search begins, for each item $i \in x^*$, its estimated frequency is initialized to $eSup_{\mathcal{D}}(i, \emptyset) = |\mathcal{V}_{\mathcal{D}}(i)|$. Once an item j has been added to the partial solution, the estimated frequencies of unbound items must be updated (see lines 4-9). Thus, we first find the variable x^{es} having the minimal estimated support (line 4).

Algorithm 3: Branching for CLOSEDDIVERSITY

```
1  In: J_max : diversity thresholds; H : history of solutions ;
2  Out: First witness index or x^es as the item with the smallest estimated support
3  begin
4      x^es ← argmin_{i∈x*}(eSup_D(i, x^+));
5      diff ← (|V_D(x^+)| − |V_D(x^+ ∪ {x^es})|);
6      foreach i ∈ x* \ {x^es} do
7          eSup_D(i, x^+ ∪ {x^es}) ← eSup_D(i, x^+) − diff;
8          if (eSup_D(i, x^+ ∪ {x^es}) < θ) then
9              eSup_D(i, x^+ ∪ {x^es}) ← |V_D(x^+ ∪ {x^es}) ∩ V_D(i)|;
10     foreach i ∈ x* do
11         if (PGrowth_UB(x^+ ∪ {i}, H, J_max)) then
12             return ⟨i, true⟩;
13     return ⟨x^es, false⟩
14 Function PGrowth_UB(x^+ ∪ {j}, H, J_max) : Boolean
15     foreach H ∈ H do
16         if (UB_J(x^+ ∪ {j}, H) > J_max) then
17             return false
18     return true
```

Next, each item $i \in x^* \setminus \{x^{es}\}$ may lose some support, but no more than $|\mathcal{V}_{\mathcal{D}}(x^+)| - |\mathcal{V}_{\mathcal{D}}(x^+ \cup \{x^{es}\})|$, since some removed transactions may not contain i (line 5). Using this upper bound (denoted by *diff*), the estimated frequency of i is updated and set to $eSup_{\mathcal{D}}(i, x^+) - \textit{diff}$ (lines 6-9). As indicated above, if $eSup_{\mathcal{D}}(i, x^+) \geq \theta$ then $|\mathcal{V}_{\mathcal{D}}(x^+ \cup \{i\})| \geq \theta$. Otherwise, we have to compute the right support by performing the intersection between covers (line 9). It is important to stress that the branching variable x^{es} will be returned (line 13) only if no positive witness is found (lines 10-12). Finally, the function $\mathcal{PG}rowth_{UB}(x^+ \cup \{i\}, \mathcal{H}, J_{max})$ allows to test whether the current instantiation x^+ can be extended to a witness itemset using the free item $\{i\}$. It returns true if the upper bound of the current itemset x^+ when adding one item $\{i\}$ is less than J_{max} for all $h \in \mathcal{H}$ (lines 15-17). Here, the Jaccard constraint is fully satisfied and thus, we return the item $\{i\}$ with the witness flag set to *true*. This information will be supplied to the search engine (line 12) to accelerate solutions certification. We will denote by FIRSTWITCOV, our variable ordering heuristic that branches on the first free item satisfying the witness property.

Exploring the witness subtree. Let N be the node associated to the current itemset x^+ extended to a free item $\{i\}$. When the node N is detected as a positive witness during the branching, all supersets derived from N will also satisfy the Jaccard constraint. As these patterns are more likely to have similar covers, so a rather high Jaccard between them, we propose a simple strategy which avoids a complete exploration of the witness sub-tree rooted at N. Thus, we generate the first closed diverse itemset from N, add it to the current history and continue the exploration of the remaining search space using the new history. With a such strategy we have no guarantee that the closed itemset added to the history have the best Jaccard. But this strategy is fast.

5 Related work

The question of mining sets of diverse patterns has been addressed in the recent literature, both to offer more interesting results and to speed up the mining process. Van Leeuwen *et al.* propose populating the beam for subgroup discovery not purely with the *best* partial patterns to be extended but to take coverage overlap into account [20]. Beam search is heuristic, as opposed to our exhaustive approach and since they mine all patterns at the same time, diverse partial patterns can still lead to a less diverse final result. Dzyuba *et al.* propose using *XOR* constraints to partition the transaction set into disjoint subsets that are small enough to be efficiently mined using either a CP approach or a dedicated itemset miner [8]. Their focus is on efficiency, which they demonstrate by approximating the result set of an exhaustive operation. While they discuss pattern sets, they limit themselves to a strict non-overlap constraint on coverages. In [4], the authors propose using Monte Carlo Tree Search and upper confidence bounds to direct the search towards interesting regions in the lattice given the already explored space. While MCTS is necessarily randomized, it allows for anytime mining. The authors of [3] consider sets of subgroup descriptions as *disjunctions* of such patterns. Using a greedy algorithm exploiting upper bounds, the authors propose to iteratively extract up to k subgroup descriptions (similarly to our work). Notably, this approach requires a target attribute *and* a target value to focus on while our approach allows for unsupervised mining.

Earlier work has treated reducing redundancy as a post-processing step, e.g. [12] where a number of redundancy measures such as entropy are exploited in exhaustive search and the number of patterns in the set limited, [7] where the constraint-based itemset mining constraint is adapted to the pattern set settings, [5], which exploit bounds on predicting the presence of patterns from the patterns already included in $\mathcal{H}$ in a heuristic algorithm, or [21], which exploits the MDL principle to minimize redundancy among itemsets (and, in later work, sequential patterns). All of those methods require a potentially rather costly first mining step, and none exploits the Jaccard measure. As discussed in Section 2.1, there exist a number of constraint properties that allow for pruning, and Kifer *et al.*'s witness concept unifies them and discusses how to deal with constraints that do not have monotonicity properties [11]. The way to proceed in such a case is establishing *positive* and *negative* witnesses for the constraint, something we have done for the maximum pairwise Jaccard constraint. A rarely discussed aspect is that witnesses are closely related to CP since every witness enforces/forbids the inclusion of certain domain values.

6 Experiments and Results

The experimental evaluation is designed to address the following questions: (1) How (in terms of CPU-times and # of patterns) does our global constraint (denoted ClosedDiv) compare to the ClosedPatterns global constraint (denoted ClosedP) and the approach of Dzyuba et al. [8] (denoted Flexics)? (2) How do the resulting diverse FCIs compare qualitatively with those resulting from ClosedP and Flexics? (3) How far is the distance between the Jaccard index and the upper/lower bounds.

Dataset $\lvert\mathcal{I}\rvert \times \lvert\mathcal{T}\rvert$ $\rho(\%)$	$\theta(\%)$	#Patterns (1)	#Patterns (2)	Time (s) (1)	Time (s) (2)	#Nodes (2)	#Nodes (2)
CHESS	20	22,808,625	**96**	2838.30	**5.87**	45,617,249	**436**
75 × 3196	15	50,723,131	**393**	5666.03	**75.40**	101,446,261	**1,855**
49.33%	10	OOM	**4,204**	OOM	**3825.29**	OOM	**18,270**
HEPATITIS	30	83,048	**12**	9.64	**0.09**	166,095	**29**
68 × 137	20	410,318	**57**	42.00	**0.57**	820,635	**162**
50.00%	10	1,827,264	**2,270**	**169.59**	**76.91**	3,654,527	**5,256**
KR-VS-KP	30	5,219,727	**17**	682.94	**0.74**	10,439,453	**82**
73 × 3196	20	21,676,719	**96**	2100.79	**5.64**	43,353,437	**448**
49.32%	10	OOM	**4,120**	OOM	**3035.49**	OOM	**17,861**
CONNECT	30	460,357	**18**	1666.14	**14.81**	920,713	**77**
129 × 67557	18	2,005,476	**197**	5975.44	**573.66**	4,010,951	**900**
33.33%	15	3,254,780	**509**	9534.07	**1989.35**	6,509,559	**2,188**
HEART-CLEVELAND	10	12,774,456	**3,496**	1308.63	**257.39**	25,548,911	**7,977**
95 × 296	8	23,278,687	**12,842**	**2278.97**	2527.38	46,557,373	**28,221**
47.37%	6	43,588,346	58,240	**4126.84**	46163.06	87,176,691	124,705
SPLICE1	10	1,606	**422**	**6.55**	25.25	3,211	**843**
287 × 3190	5	31,441	**8,781**	**117.15**	5616.47	62,881	**17,594**
20.91%	2	589,588	-	**1179.55**	-	1,179,175	-

Dataset $\lvert\mathcal{I}\rvert \times \lvert\mathcal{T}\rvert$ $\rho(\%)$	$\theta(\%)$	#Patterns (1)	#Patterns (2)	Time (s) (1)	Time (s) (2)	#Nodes (2)	#Nodes (2)
MUSHROOM	5	8,977	**727**	**10.02**	60.70	17,953	**1,704**
112 × 8124	1	40,368	**12,139**	**34.76**	12532.95	80,735	**25,154**
18.75%	0.5	62,334	**27,768**	**50.05**	64829.06	124,667	**56,873**
T40I10D100K	8	138	127	**75.91**	447.20	275	253
942 × 100000	5	317	288	**331.47**	1561.34	633	575
4.20%	1	65,237	7,402	**5574.31**	58613.88	130,473	14,887
PUMSB	40	-	**4**	-	**57.33**	-	**16**
2113 × 49046	30	-	**15**	-	**267.72**	-	**64**
3.50%	20	-	**52**	-	**852.39**	-	**250**
T10I4D100K	5	11	11	**1.73**	6.31	21	21
870 × 100000	1	386	**361**	**434.25**	3125.06	771	722
1.16%	0.5	1,074	**617**	**881.31**	7078.90	2,147	**1,257**
BMS1	0.15	1,426	**609**	**11362.71**	68312.38	2,851	**1,220**
497 × 59602	0.14	1,683	**668**	**11464.93**	68049.00	3,365	**1,339**
0.51%	0.12	2,374	**823**	**13255.79**	79704.88	4,747	**1,651**
RETAIL	5	17	**13**	**10.74**	33.44	33	25
16470 × 88162	1	160	**111**	**297.21**	1625.73	319	227
0.06%	0.4	832	**528**	**6073.53**	31353.23	1,663	**1,093**

Table 1: CLOSEDDIV ($J_{max} = 0.05$) vs CLOSEDP. For columns #Patterns and #Nodes, the values in bold indicate a reduction more than 20% of the total number of patterns and nodes." − " is shown when time limit is exceeded. OOM : Out Of Memory. (1): CLOSEDP (2): CLOSEDDIV

Experimental protocol. Experiments were carried out on classic UCI data sets, available at the FIMI repository (fimi.ua.ac.be/data). We selected several real-world data sets, their characteristics (name, number of items $|\mathcal{I}|$, number of transactions $|\mathcal{T}|$, density ρ) are shown in the first column of Table 1. We selected data sets of various size and density. Some data sets, such as Hepatitis and Chess, are very dense (resp. 50% and 49%). Others, such as T10 and Retail, are very sparse (resp. 1% and 0.06%). The implementation of the different global constraints and their constraint propagators were carried out in the `Choco` solver [17] version $4.0.3$, a Java library for constraint programming. The source code is publicly available.[5] Experiments were conducted on AMD Opteron 6174, 2.2 GHz with a RAM of 256 GB and a time limit of 24 hours. The default maximum heap size allowed by the JVM is 30 GB. We have selected for every data set frequency thresholds to have different numbers of frequent closed itemsets ($|Th(c)| \leq 15000$, $30000 \leq |Th(c)| \leq 10^6$, and $|Th(c)| > 10^6$). The only exception are the very large and sparse data sets Retail and Pumsb, where we do not find a large number of solutions. We used the CLOSEDP CP model as a baseline to determine suitable thresholds used with the CLOSEDDIV CP model. To evaluate the quality of a set of patterns in terms of diversity, we measured the average ratio of exclusive pattern coverages: $ECR(P_1, ..., P_k) = avg_{1 \leq i \leq k}(\frac{sup_{\mathcal{D}}(P_i) - |\mathcal{V}_{\mathcal{D}}(P_i) \cap \bigcup_{j \neq i} \mathcal{V}_{\mathcal{D}}(P_j)|}{sup_{\mathcal{D}}(P_i)})$.

(a) Comparing CLOSEDDIV with CLOSEDP and FLEXICS. Table 1 compares the performance of the two CP models for various values of θ on different data sets. Here, we report the CPU time (in seconds), the number of extracted patterns, and the number of nodes explored during search. This enables to evaluate the amount of inconsistent values pruned by each approach (filtering algorithm). We use MINCOV as variable ordering heuristic. The maximum diversity threshold J_{max} is set to 0.05. First, the results highlight the great discrepancy between the two models with a distinctly lower number of patterns generated by CLOSEDDIV (in the thousands) in comparison to CLOSEDP (in the millions). On dense and moderately dense data sets (from CHESS to MUSHROOM),

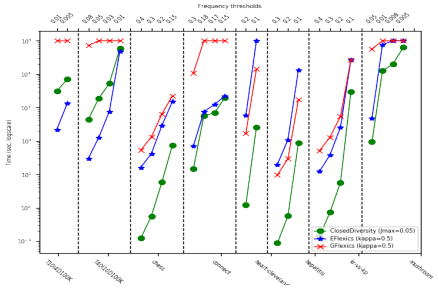
[5] https://github.com/lobnury/ClosedDiversity

the discrepancy is greatly amplified, especially for small values of θ. For instance, on CHESS, the number of patterns for CLOSEDDIV is reduced by 99% (from $\sim 50 \cdot 10^6$ solutions to 393) for θ equal to 15%. The density of the data sets provides an appropriate explanation for the good performance of CLOSEDDIV. As the number of closed patterns increases with the density, redundancy among these patterns increases as well. On very sparse data sets, CLOSEDDIV still outputs fewer solutions than CLOSEDP but the difference is less pronounced. This is explainable by the fact that on these data sets, where we have few solutions, almost all patterns are diverse.

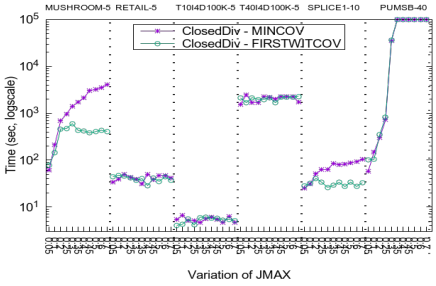
Second, regarding runtime, CLOSEDDIV exhibits different behaviours. On dense data sets ($\rho \geq 30\%$), CLOSEDDIV is more efficient than CLOSEDP and up to an order of magnitude faster. On CHESS (resp. CONNECT), the speed-up is 1455 (resp. 112) for $\theta = 30\%$. For instances resulting in between 500 and 5000 diverse FCIs, the speed-up is up to 5. This good performance of CLOSEDDIV is mainly due to the strength of the LB filtering rule that provides the CP solving process with more propagation to remove more inconsistent values in the search space. In addition, the number of nodes explored by CLOSEDDIV is always small comparing to CLOSEDP. These results support our previous observations. The only exception is HEART-CLEVELAND for which CLOSEDDIV is slower (especially for values of $\theta \leq 8\%$). This is mainly due to the relative large number of diverse patterns (≥ 12000), which induces higher lower bound computational overhead. We observe the same behaviour on the two moderately dense data sets SPLICE1 and MUSHROOM. On sparse data sets, CLOSEDDIV can take significantly more time to extract all diverse FCIs. This can be explained by the fact that on these instances almost all FCIs are diverse w.r.t. lower bound (on average about 70% for RETAIL and 39% for BMS1, see Table 1). Thus, non-solutions are rarely filtered, while the lower bound overhead greatly penalizes the CP solving process. On the very large PUMSB data set, finally, our approach is very efficient while CLOSEDP fails to complete the extraction.

Finally, Figure 2a compares CLOSEDDIV with FLEXICS (two variants) for various values of θ on different data sets: GFLEXICS, which uses CP4IM [6] as an oracle to enumerate the solutions, and EFLEXICS, a specialized variant, based on ECLAT [23]. We run WEIGHTGEN with values of $\kappa \in \{0.1, 0.5, 0.9\}$ [9]. For each instance, we fixed the number of samples to the number of solutions returned by CLOSEDDIV. We report results corresponding to the best setting of parameter κ. First, CLOSEDDIV largely dominates GFLEXICS, being more than an order of magnitude faster. Second, while EFLEXICS is faster than GFLEXICS, our approach is almost always ranked first, illustrating its usefulness for mining diverse patterns in an anytime manner.

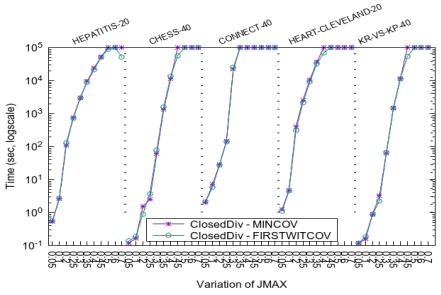
(b) Impact of varying J_{max}. We varied J_{max} from 0.1 to 0.7. The minimum support θ is fixed for each data set (indicated after '-'). Figure 2 shows detailed results. As expected, the greater J_{max}, the longer the CPU time. In fact, the size of the history $\mathcal{H}$ grows rapidly with the increase of J_{max}. This induces significant additional costs in the lower and upper bound computations. Moreover, when J_{max} becomes sufficiently large, the LB filtering of CLOSEDDIV occurs rarely since the lower bound is almost always below the J_{max} value (see Figure 3). Despite the hardness of some instances (i.e., $J_{max} \geq 0.35$), our CP approach is able to complete the extraction for almost all values of J_{max}. The only exception are the large and dense data sets, where CLOSEDDIV fails



(a) CPU-time MINCOV and FLEXICS



(b) Moderately dense and sparse dat sets.



(c) Dense data sets.

Dataset	θ(%)	Exclusive Coverage Ratio (ECR)		
		CLOSEDP	CLOSEDDIV–MINCOV	EFLEXICS
T40I10D100K	8	1.6E-01	**4.4E-01**	1.29E-01
	1	1.9E-01	**9.2E-01**	2.90E-01
SPLICE1	10	4.0E-02	**1.6E-01**	2.98E-02
	5	4.0E-02	**2.5E-01**	1.79E-01
CONNECT	18	0.00	**1.0E-01**	4.31E-02
SPLICE1	10	4E-02	**1.6E-01**	2.98E-02
	5	4E-2	**2.5E-01**	1.79E-01
CONNECT	18	1.25E-6	**1.0E-01**	4.31E-02
	15	0.00	**1.7E-01**	4.16E-02
MUSHROOM	5	3.3E-4	**5.6E-01**	2.63E-01
	1	3.73E-4	**4.3E-01**	4.2E-01
HEPATITIS	10	0.00	**2.5E-01**	1.0E-01
HEPATITIS	10	1.50E-3	**2.5E-01**	1.0E-01
T10I4D100K	1	5.9E-01	**9.3E-01**	5.72E-01
	0.5	4.8E-01	**9.6E-01**	4.63E-01

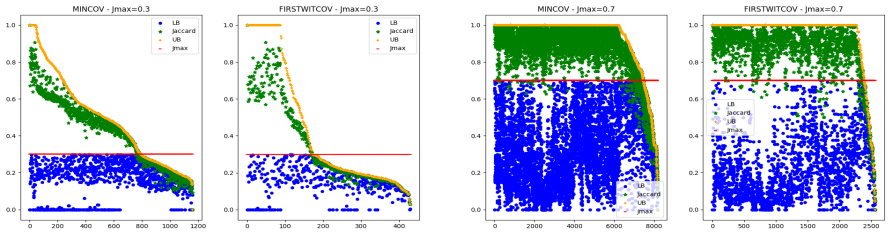
(d) Patterns ECR.

Fig. 2: CPU-time analysis (MINCOV vs FIRSTWITCOV and CLOSEDDIV vs FLEXICS) and patterns discrepancy analysis.

to complete the extraction within the time limit for $J_{max} \geq 0.45$. However, in practice, the user will only be interested in small values of J_{max} because the diversity of patterns is maximal and the number of patterns returned becomes manageable.

Figures 2b and 2c also compare the resolution time of our CP model using the two variable ordering heuristics MINCOV and FIRSTWITCOV. First, on dense data sets, both heuristics perform similarly, with a slight advantage for FIRSTWITCOV. On these data sets, the number of witness patterns mined remains very low ($\leq$ 100), thus the benefits of FIRSTWITCOV is limited (see Supp. material). On moderately dense data sets (MUSHROOM and SPLICE1), FIRSTWITCOV is very effective; on MUSHROOM it is up 10 times faster than MINCOV for J_{max} equal to 0.7. On these data sets, the number of witness patterns extracted is relatively high compared to dense ones. In this case, FIRSTWITCOV enables to guide the search to find diverse patterns more quickly. On sparse data sets, no heuristic clearly dominates the other. When regarding the number of diverse patterns generated (see Supp. material), we observe that FIRSTWITCOV returns less patterns on moderately dense and sparse data sets, while on dense data sets the number of diverse patterns extracted remains comparable.

(c) Qualitative analysis of the proposed relaxation. In this section, we shed light on the quality of the relaxation of the Jaccard constraint. Figure 3a shows, for a particular instance SPLICE1 with $J_{max} = 0.3$, the evolution of the LB_J and UB_J of the solutions



(a) SPLICE1 ($\theta = 10\%$, $J_{max} = 0.3$) (b) MUSHROOM ($\theta = 5\%$, $J_{max} = 0.7$)

Fig. 3: Qualitative analysis of the LB and UB relaxations.

found during search. Here, the solutions are sorted according to their UB_J. Concerning the lower bound, one can observe that the LB_J values are always below the J_{max} value. This shows how frequently the LB filtering rule of CLOSEDDIV occurs. This also supports the suitability of the LB filtering rule for pruning non-diverse FCIs. With regard to the upper bound, it is interesting to see that it gets very close to the Jaccard value, meaning that our Jaccard upper bounding provides a tight relaxation. Moreover, a large number of solutions have UB_J values either below or very close to J_{max}. This is indicative of the quality of the patterns found in terms of diversity. We recall that when $UB_J < J_{max}$, all partial assignments can immediately be extended to diverse itemsets, thanks to the anti-monotonicity property of our UB (see Proposition 5). We observe the same behaviour on MUSHROOM with $J_{max} = 0.7$ (see Figure. 3b). Finally, we can see that FIRSTWITCOV allows to quickly discover solutions of better quality in terms of UB_J and Jaccard values compared to MINCOV. This demonstrates the interest and the strength of our UB_J branching rule to get diverse patterns.

(d) Qualitative analysis of patterns. Figure 2d compares CLOSEDDIV with CLOSEDP and EFLEXICS in terms of the ECR measure, which should be as high as possible. Due to the huge number of patterns generated by CLOSEDP, a random sample of $k = 10$ solutions of all patterns is considered. Reported values are the average over 100 trials. ECR penalises overlap, and thus having two similar patterns is undesirable. According to ECR, leveraging Jaccard in CLOSEDDIV clearly leads to pattern sets with more diversity among the patterns. This is indicative of patterns whose coverage are (approximately) mutually exclusive. This should be desirable for an end-user tasked with exploring and interpreting the set of returned patterns.

7 Conclusions

In this paper, we showed that mining diverse patterns using a maximum Jaccard constraint cannot be modeled using an anti-monotonic constraint. Thus, we have proposed (anti-)monotonic lower and upper bound relaxations, which allow to make pruning effective, with an efficient branching rule, boosting the whole search process. The proposed approach is introduced as a global constraint called CLOSEDDIV where diversity is controlled through a threshold on the Jaccard similarity of pattern occurrences. Experimental results on UCI datasets demonstrate that our approach significantly reduces

the number of patterns, the set of patterns is diverse and the computation time is lower compared to CLOSEDP global constraint, particularly on dense data sets.

References

1. Supplementary Material (June 2020), https://github.com/lobnury/ClosedDiversity.
2. Belaid, M., Bessiere, C., Lazaar, N.: Constraint programming for mining borders of frequent itemsets. In: Proceedings of IJCAI 2019, Macao, China. pp. 1064–1070 (2019)
3. Belfodil, A., Belfodil, A., Bendimerad, A., Lamarre, P., Robardet, C., Kaytoue, M., Plantevit, M.: Fssd-a fast and efficient algorithm for subgroup set discovery. In: Proceedings of DSAA. pp. 91–99 (2019)
4. Bosc, G., Boulicaut, J.F., Raïssi, C., Kaytoue, M.: Anytime discovery of a diverse set of patterns with monte carlo tree search. Data mining and knowledge discovery **32**(3), 604–650 (2018)
5. Bringmann, B., Zimmermann, A.: The chosen few: On identifying valuable patterns. In: Proceedings of ICDM 2007. pp. 63–72
6. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for itemset mining. In: 14th ACM SIGKDD. pp. 204–212 (2008)
7. De Raedt, L., Zimmermann, A.: Constraint-based pattern set mining. In: 7th SIAM SDM. pp. 237–248. SIAM (2007)
8. Dzyuba, V., van Leeuwen, M., De Raedt, L.: Flexible constrained sampling with guarantees for pattern mining. Data Mining and Knowledge Discovery **31**(5), 1266–1293 (2017)
9. Dzyuba, V., van Leeuwen, M.: Interactive discovery of interesting subgroup sets. In: International Symposium on Intelligent Data Analysis. pp. 150–161. Springer (2013)
10. Hoeve, W., Katriel, I.: Global constraints. In: Handbook of Constraint Programming, pp. 169–208. Elsevier Science Inc. (2006)
11. Kifer, D., Gehrke, J., Bucila, C., White, W.: How to quickly find a witness. In: Constraint-Based Mining and Inductive Databases. pp. 216–242. Springer Berlin Heidelberg (2006)
12. Knobbe, A.J., Ho, E.K.: Pattern teams. In: Proceedings of ECML-PKDD. pp. 577–584. Springer (2006)
13. Lazaar, N., Lebbah, Y., Loudni, S., Maamar, M., Lemière, V., Bessiere, C., Boizumault, P.: A global constraint for closed frequent pattern mining. In: Proceedings of the 22nd CP. pp. 333–349 (2016)
14. van Leeuwen, M.: Interactive Data Exploration Using Pattern Mining, pp. 169–182. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
15. Ng, R.T., Lakshmanan, L.V.S., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained association rules. In: Proceedings of ACM SIGMOD. pp. 13–24 (1998)
16. Pei, J., Han, J., Lakshmanan, L.V.S.: Mining frequent item sets with convertible constraints. In: Proceedings of ICDE. pp. 433–442 (2001)
17. Prud'homme, C., Fages, J.G., Lorca, X.: Choco Solver Documentation (2016)
18. Puolamäki, K., Kang, B., Lijffijt, J., De Bie, T.: Interactive visual data exploration with subjective feedback. In: Proceedings of ECML PKDD. pp. 214–229. Springer (2016)
19. Schaus, P., Aoga, J.O.R., Guns, T.: Coversize: A global constraint for frequency-based itemset mining. In: Proceedings of the 23rd CP 2017. pp. 529–546 (2017)
20. Van Leeuwen, M., Knobbe, A.: Diverse subgroup set discovery. Data Mining and Knowledge Discovery **25**(2), 208–242 (2012)
21. Vreeken, J., Van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Mining and Knowledge Discovery **23**(1), 169–214 (2011)

22. Wang, J., Han, J., Pei, J.: CLOSET+: searching for the best strategies for mining frequent closed itemsets. In: Proceedings of the Ninth KDD. pp. 236–245. ACM (2003)
23. Zaki, M., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. In: Proceedings of KDD 1997, Newport Beach, California, USA, August 14-17, 1997. pp. 283–286. AAAI Press (1997)