Data Min Knowl Disc (2013) 26:574–619
DOI 10.1007/s10618-012-0284-8

Closed and noise-tolerant patterns in n-ary relations

Loïc Cerf · Jérémy Besson · Kim-Ngan T. Nguyen · Jean-François Boulicaut

Received: 24 March 2011 / Accepted: 1 August 2012 / Published online: 21 August 2012

Abstract Binary relation mining has been extensively studied. Nevertheless, many interesting 0/1 data naturally appear as n-ary relations with $n \geq 3$. A timely challenge is to extend local pattern extraction, e. g., closed pattern mining, to such contexts. When considering higher arities, faint noise affects more and more the quality of the extracted patterns. We study a declarative specification of error-tolerant patterns by means of new primitive constraints and the design of an efficient algorithm to extract every solution pattern. It exploits the enumeration principles of the state-of-the-art Data-Peeler algorithm for n-ary relation mining. Efficiently enforcing error-tolerance crucially depends on innovative strategies to incrementally compute partial information on the data. Our prototype is tested on both synthetic and real datasets. It returns relevant collections of patterns even in the case of noisy ternary or 4-ary relations, e. g., in the context of pattern discovery from dynamic networks.

Responsible editor: M. J. Zaki.

L. Cerf (✉)
Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
e-mail: lcerf@dcc.ufmg.br

J. Besson
University of Vilnius, Vilnius, Lithuania
e-mail: contact.jeremy.besson@gmail.com

K.-N. T. Nguyen · J.-F. Boulicaut
Université de Lyon, CNRS, INSA, LIRIS, UMR5205, 69621 Lyon, France
e-mail: tknnguyen@insa-lyon.fr

J.-F. Boulicaut
e-mail: jean-francois.boulicaut@insa-lyon.fr

Keywords Boolean tensor · Multi-way set · Fault-tolerance · Cross-graph quasi-clique

1 Introduction

Closed set mining from Boolean matrices and thus binary relations has been extensively studied since the seminal paper (Pasquier et al. 1999). A closed itemset in a given Boolean matrix that encodes whether some objects (say transactions) satisfy or not some properties (say contain or not some items) correspond to maximal sets of properties that are shared by a given set of objects. For instance if, in a given dataset, $P = \{B, C, D\}$ is a closed set associated with the supporting set of objects $O = \{1, 2, 3\}$, it means that we cannot add any property to P which would be true for all the objects in O. Notice that studying collections of such couples of sets is also known as Formal Concept Analysis (Ganter et al. 2005).

The closed set mining task has been studied in depth for two main reasons. First, thanks to the formal properties of closedness, it enables the computation of every frequent itemset in many real-life dense and correlated datasets where other techniques fail for the selected frequency thresholds. In other terms, the availability of efficient closed set mining algorithms is a breakthrough with respect to frequent itemset mining computational feasability (see, e. g., (Bayardo et al 2004; Goethals 2010)). Next, the intrinsic maximality property of closed sets has been useful for deriving more relevant patterns. Indeed, thanks to the closedness properties, it is possible to adress important issues like, for instance, the non redundancy of association rules (see, e. g., (Zaki 2004)), the computation of a priori relevant local patterns or biclusters with nice applications to gene expression analysis (see, e. g., (Pan et al. 2003)), or the construction of better features for supervized classification (see, e. g., (Garriga et al. 2008)). In fact, closedness provides a *lossless condensation* (Boulicaut and Bykowski 2000; Calders et al 2005) of all itemsets by only keeping the *most informative* ones (Gallo et al. 2007, 2009). In other terms, we know how to compute the exhaustive collection of the closed sets that hold in a given dataset, possibly enforcing user-defined constraints (see, e. g., (Stumme et al. 2002; Bonchi and Lucchese 2004; Besson et al. 2005)), and this has been proved useful across many different 0/1 data mining workflows.

The starting point of our study is that many interesting 0/1 datasets correspond to n-ary relations with $n \geq 3$ and not just binary ones. For instance, we may consider ternary relations that encode whether some objects (first attribute) satisfy some properties (second attribute) at given timestamps (third attribute). Therefore, a timely challenge is to extend pattern discovery techniques from binary to arbitrary n-ary relations and thus to Boolean tensors instead of just Boolean matrices. It makes sense to look for closed patterns in such an extended setting. Considering that formal concepts correspond to the so-called closed 2-sets in binary relations, recent algorithms like CubeMiner (Ji et al. 2006) and Trias (Jaschke et al. 2006) extract every closed pattern when $n = 3$, i. e., closed 3-sets. Data- Peeler (Cerf et al. 2009a) efficiently computes closed patterns for an arbitrary arity n, i. e., closed n-sets.

A major bottleneck for the dissemination of data mining methods based on closed pattern discovery is related to noise and its impact on the computed collections. Real

n-ary relations suffer from noise that can have several causes (e. g., intrinsic noise in the studied system, measurement errors, chosen discretization thresholds, etc.). As a result, some n-tuples present in (resp. absent from) the relation should, in fact, be absent (resp. present). Minimal size constraints (i. e., looking for large enough patterns) allow to avoid the n-tuples *present* in the relation because of noise. On the contrary, when noise *drops* some n-tuples, not only the number of closed patterns increases but their relevancy is affected too. For example, assume a binary relation in which $\{B, C, D\}$ is a closed itemset whose support (i. e., all transactions sharing the items B, C and D) is $\{1, 2, 3\}$. By definition, $(\{1, 2, 3\}, \{B, C, D\})$ is a closed 2-set. Let us now assume that, because of noise, $(2, C)$ is absent from the relation. Instead of $(\{1, 2, 3\}, \{B, C, D\})$, two other patterns, namely $(\{1, 3\}, \{B, C, D\})$ and $(\{1, 2, 3\}, \{B, D\})$, are extracted. In fact, an exponential growth of the number of patterns occurs and the extracted closed 2-sets are less relevant since they only are logarithmic-size fragments of the "real" patterns (Liu et al. 2006). To extract them, there is a need to relax the minimal size constraints and some other patterns, that are really too small to be relevant (e. g., including 2-tuples that are present in the relation because of noise), are found as well. These phenomena are dramatically aggravated when mining relations of higher arities. Indeed, since the number of n-tuples inside a closed n-set exponentially increases with n, noise has a greater probability to damage one (or several) of its n-tuples. To address such problems, we generalize the definition of a closed n-set to let it encompass, within user-defined bounds, some absent n-tuples considered as noise. Different definitions were proposed for itemset mining in binary relations (see Gupta et al. (2008) for a survey). Such a declarative specification for a *closed ET-n-set*[1] remains open.

We present a definition of noise tolerance that is applicable to patterns in arbitrary n-ary relations. It relies on upper-bounded quantities of noise that are tolerated per element in the pattern. For instance, in our running example, $(\{1, 2, 3\}, \{B, C, D\})$ would be discovered, despite the absence of $(2, C)$ from the relation, if the analyst decides to tolerate one error per element (2 and C reach this bound, 1, 3, B and D do not encompass any noise). Unlike most of the literature on noise-tolerant mining in binary relations, our definition of a noise tolerant pattern includes a *closedness* constraint. In this way, redundant patterns are avoided. More precisely, the closedness constraint filters the most informative patterns without any loss of information w.r.t. the collection that would be obtained without this constraint. Designing an algorithm to extract every such pattern (*completeness*) while remaining *scalable* is challenging. This article describes such an algorithm, namely FENSTER[2]. Part of its efficiency can be attributed to enumeration principles that it largely borrows from the state-of-the-art closed n-set extractor DATA- PEELER (Cerf et al. 2009a). These same principles also allow the efficient enforcement of additional constraints (instances of one of the most expressive class defined so far) that makes FENSTER able to discover specific patterns

[1] Like in Yang et al. (2000), ET stands for Error-Tolerant. In the text, the more general term "noise" is preferred to "error". Indeed, the noise is not necessarily caused by errors and it can be intrinsic to the studied system, for instance in life sciences.

[2] FENSTER stands for FENSTER Extracts N-Sets Tolerating Errors in the Relation.

in large datasets. The discovery of maximal quasi-cliques in a labeled, dynamic and directed transportation network is a real-life example detailed in Sect. 6.

Although the enumeration principles are essential, tolerating noise significantly harden the algorithmic challenge. To remain scalable, FENSTER incrementally updates counts of absent n-tuples in various subspaces of the data. Thanks to these aggregates, the same regions of the relation are not browsed again and again to check whether the recursively refined search space still contains a pattern satisfying the definition. In fact, we theoretically show that FENSTER's enforcement of the definition is as fast as a naive one in a relation with the same number of elements per dimension but one dimension less. An empirical comparison with DCE, the only other algorithm able to extract noise tolerant patterns in arbitrary n-ary relation[3], confirms the good performance of FENSTER. Indeed, our proposal turns out to be orders of magnitude faster. This comparison validates empirically FENSTER's per-element tolerance to noise: the quality of the patterns extracted under DCE's per-pattern tolerance to noise is both lower than that of FENSTER and less robust, i. e., more sensitive to the chosen parametrization.

The next section introduces the closed ET-n-set mining task. The enumeration principles of FENSTER are sketched in Sect. 3. The crucial implementation details are given in Sect. 4, which discusses as well the time and space complexities. Section 5 deals with an empirical validation on synthetic datasets, whereas Sect. 6 presents results on a real graph evolving in two temporal dimensions. Section 7 is dedicated to related work and Sect. 8 briefly concludes.

2 Definitions and problem setting

2.1 N-ary relations as data

Given an arity $n \in \mathbb{N}$ and n finite sets $(\mathcal{D}^i)_{i=1..n}$, let $\mathcal{R} \subseteq \times_{i=1..n}\mathcal{D}^i$ ($\times_{i=1..n}\mathcal{D}^i$ stands for $D^1 \times D^2 \times \cdots \times D^n$) the n-ary relation where patterns are to be discovered. All along this article, $\mathcal{R}$ denotes this dataset. Table 1 represents an example of such a relation $\mathcal{R}_E \subseteq \{\alpha, \beta, \gamma\} \times \{1, 2, 3, 4\} \times \{A, B, C\}$, hence a ternary relation. In this table, every '1' at the intersection of three elements stands for the presence of the related triplet in $\mathcal{R}_E$. For example the bold '**1**' at the intersection of the elements α, 1 and A means $(\alpha, 1, A) \in \mathcal{R}_E$. On the contrary a '0' in Table 1 is at the intersection of three elements that form a triplet absent from $\mathcal{R}_E$. For example the bold '**0**'means $(\alpha, 2, C) \notin \mathcal{R}_E$. N-ary relations are available in many application domains. For instance, $\mathcal{R}_E$ could represent customers (1, 2, 3 and 4) buying items (A, B and C) along three months (α, β and γ). In this context, the bold '**1**' in Table 1 would mean that "Customer 1 bought Item A during the first month". The bold '**0**' would be understood as "Customer 2 did not buy Item C during the first month".

More generally, binary relation mining has been quite popular in the many application domains where we can record Boolean properties of sets of Objects, i. e., mining

[3] DCE actually is applicable in even more general contexts where the n-tuples can be associated with degrees of certainty.

Table 1 $\mathcal{R}_E \subseteq \{\alpha, \beta, \gamma\} \times \{1, 2, 3, 4\} \times \{A, B, C\}$

	A	B	C	A	B	C	A	B	C
1	**1**	1	1	1	1	1	1	1	0
2	1	1	**0**	1	0	0	1	1	0
3	0	1	0	0	0	1	1	0	1
4	0	0	1	1	0	1	1	1	1
		α			β			γ	

an $Objects \times Properties$ relation. A generic though common context can be to add spatial and/or time information to get an $Objects \times Properties \times Dates$ and/or an $Objects \times Properties \times Dates \times Places$ relations. Among others, we are interested by dynamic graph description by means of n-ary relations. The idea is that the two first dimensions correspond to the vertices such that we encode the graph incidence matrix. Given $\mathcal{R}_E$ could represent vertices (1, 2, 3 and 4) which can be linked or not to the vertices (A, B and C) along three time-stamps (α, β and γ). In this context, the bold '**1**' in Table 1 would mean Vertex 1 is connected to Vertex A at the first time-stamps. Section 6 is a case study on a real graph which evolves in two temporal dimensions.

2.2 Closed n-sets as patterns

The patterns in $\times_{i=1..n} 2^{\mathcal{D}^i}$ are called n-sets. They associate n subsets of elements from the n domains of the relation. For the sake of clarity, an n-set $(S^i)_{i=1..n}$ will often be represented by $\cup_{i=1..n} S^i$ (without loss of generality, the attribute domains $\mathcal{D}^i$ are considered disjoint). For example, given an n-set $S = (S^i)_{i=1..n}$ and an element $e \in \cup_{i=1..n} \mathcal{D}^i$, we write:

- $e \in S$ instead of $e \in \cup_{i=1..n} S^i$;
- $S \setminus \{e\}$ instead of $\begin{cases} (S^1 \setminus \{e\}, S^2, \ldots, S^n) \text{ if } e \in \mathcal{D}^1 \\ \vdots \\ (S^1, \ldots, S^{n-1}, S^n \setminus \{e\}) \text{ if } e \in \mathcal{D}^n \end{cases}$.

Let us formalize the union and the inclusion on n-sets.

Definition 1 (**n-set union** $\sqcup$) Given two n-sets $S = (S^i)_{i=1..n} \in \times_{i=1..n} 2^{\mathcal{D}^i}$ and $T = (T^j)_{j=1..n} \in \times_{j=1..n} 2^{\mathcal{D}^j}$, $S \sqcup T = (S^1 \cup T^1, \ldots, S^n \cup T^n)$.

Definition 2 (**n-set inclusion** $\sqsubseteq$) Given two n-sets $S = (S^i)_{i=1..n} \in \times_{i=1..n} 2^{\mathcal{D}^i}$ and $T = (T^j)_{j=1..n} \in \times_{j=1..n} 2^{\mathcal{D}^j}$, $S \sqsubseteq T \Leftrightarrow S^1 \subseteq T^1 \wedge \cdots \wedge S^n \subseteq T^n$.

Notice that, by definition, the union of two n-sets is the n-set with the minimal envelope enclosing both of them. The inclusion of n-sets is useful to define a closedness constraint. Closed n-sets in n-ary relations ($n \geq 2$) generalize closed itemsets in binary relations: the closed 2-sets *are* the closed itemsets associated to their supporting sets of transactions or objects, i. e., the so-called formal concepts (Ganter et al. 2005). The generalization of the definition towards n-ary relations is natural. Considering the 0/1

representation of the n-ary relation (such as $\mathcal{R}_E$ in Table 1), a closed n-set is a maximal hyper-rectangle of '1's modulo arbitrary permutations of the hyperplanes (i. e., the hyperplanes of a pattern need not be contiguous).

In the following, when X denotes an n-set, we assume $X = (X^i)_{i=1..n} \in \times_{i=1..n} 2^{\mathcal{D}^i}$.

Definition 3 (Closed n-set) Given an n-set X, X is a closed n-set iff it satisfies the two following constraints:

- $\mathcal{C}_{\text{connected}}(X) \equiv \times_{i=1..n} X^i \subseteq \mathcal{R}$;
- $\mathcal{C}_{\text{closed}}(X) \equiv \forall X' \in \times_{j=1..n} 2^{\mathcal{D}^j}, \left(X \sqsubseteq X' \wedge \mathcal{C}_{\text{connected}}(X')\right) \Rightarrow X' = X.$

According to the first constraint, $\mathcal{C}_{\text{connected}}$, taking one element from each of the subsets constituting a closed n-set is constructing an n-tuple that is in $\mathcal{R}$. The second constraint, $\mathcal{C}_{\text{closed}}$, tells that X is closed if any strictly larger pattern (more elements from any domains) violates $\mathcal{C}_{\text{connected}}$. It is, for $\mathcal{C}_{\text{connected}}$, a closure property on the n subsets of $\mathcal{D}^1$, $\mathcal{D}^2$, ... and $\mathcal{D}^n$ altogether. It can easily be proved that an equivalent closedness constraint only forces the patterns with *one* more element (from any domain) to break $\mathcal{C}_{\text{connected}}$. Furthermore, because $\mathcal{C}_{\text{connected}}$ ensures the presence in $\mathcal{R}$ of every n-tuple in $\times_{i=1..n} X^i$, checking the closedness constraint can be reduced to searching for absent n-tuples involving this additional element only.

Definition 4 (Closed n-set (equivalent definition)) Given an n-set X, X is a closed n-set iff it satisfies the two following constraints:

- $\mathcal{C}_{\text{connected}}(X) \equiv \times_{i=1..n} X^i \subseteq \mathcal{R}$;
- $\mathcal{C}_{\text{closed}}(X) \equiv \forall i = 1..n, \forall s \in \mathcal{D}^i \setminus X^i,$
  $\neg\mathcal{C}_{\text{connected}}(X^1, \ldots, \{s\}, \ldots, X^n)$, i. e., $X^1 \times \cdots \times \{s\} \times \cdots \times X^n \not\subseteq \mathcal{R}$.

Example 1 In $\mathcal{R}_E$, represented in Table 1, $(\{\alpha, \gamma\}, \{1, 2\}, \{A, B\})$ is a closed 3-set:

$\{\alpha, \gamma\} \times \{1, 2\} \times \{A, B\} \subseteq \mathcal{R}_E$ (in Table 1 there are '1's at the intersection of all the related hyperplanes);
Every pattern with one more element violates $\mathcal{C}_{\text{connected}}$:
- $\neg\mathcal{C}_{\text{connected}}(\{\beta\}, \{1, 2\}, \{A, B\})$, i. e., $\{\beta\} \times \{1, 2\} \times \{A, B\} \not\subseteq \mathcal{R}_E$;
- $\neg\mathcal{C}_{\text{connected}}(\{\alpha, \gamma\}, \{3\}, \{A, B\})$, i. e., $\{\alpha, \gamma\} \times \{3\} \times \{A, B\} \not\subseteq \mathcal{R}_E$;
- $\neg\mathcal{C}_{\text{connected}}(\{\alpha, \gamma\}, \{4\}, \{A, B\})$, i. e., $\{\alpha, \gamma\} \times \{4\} \times \{A, B\} \not\subseteq \mathcal{R}_E$;
- $\neg\mathcal{C}_{\text{connected}}(\{\alpha, \gamma\}, \{1, 2\}, \{C\})$, i. e., $\{\alpha, \gamma\} \times \{1, 2\} \times \{C\} \not\subseteq \mathcal{R}_E$.

$(\{\alpha, \beta, \gamma\}, \{1, 2\}, \{A\})$ and $(\{\alpha, \beta, \gamma\}, \{1, 2, 3, 4\}, \emptyset)$ are two other examples of closed 3-sets in $\mathcal{R}_E$.

Considering, again, a ternary relation that stands for customers buying items along months, a closed 3-set is a maximal subset of customers buying the same maximal subset of items during a maximal subset of months. Such a pattern is useful for analyzing buying behaviors. The closedness constraint filters out all strict "sub-patterns" (i. e., patterns where some elements are removed and none are added) of the largest ones that are extracted. The justification for this constraint is the same as with itemset mining: a lossless (Boulicaut and Bykowski 2000) reduction of the output, which

keeps the most informative pattern of every equivalence class (Gallo et al. 2009). With collections of n-sets, a "lossless condensation" means that, whatever $j = 1..n$ and given any $(n-1)$-set $X \in \times_{i=1..n \wedge i \neq j} 2^{\mathcal{D}^i}$, *all* elements in $\mathcal{D}^j$ that relate with every combination of $n-1$ elements taken from the $n-1$ subsets of X can be derived from the closed n-sets only. They are the largest set of elements in $\mathcal{D}^j$ a closed n-set associates with an $(n-1)$-set larger than X (w.r.t. the $\sqsubseteq$ order).

2.3 Absolute noise-tolerance

The definition of a closed n-set is too strict to enable the discovery of relevant patterns in *noisy* n-ary relations. The definition of a closed ET-n-set relaxes that of a closed n-set. It is based on absolute noise-tolerance parameters $\epsilon = (\epsilon^i)_{i=1..n} \in \mathbb{N}^n$. Given ϵ, the type of pattern that is to be mined is defined by a conjunction of two constraints, namely $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$.

Definition 5 **($\mathcal{C}_{\epsilon\text{-connected}}$)** Given an n-set X, $\mathcal{C}_{\epsilon\text{-connected}}(X) \equiv \forall i = 1..n, \forall e \in X^i, |(X^1 \times \cdots \times \{e\} \times \cdots \times X^n) \setminus \mathcal{R}| \leq \epsilon^i$.

Definition 6 **($\mathcal{C}_{\epsilon\text{-closed}}$)** Given an n-set X, $\mathcal{C}_{\epsilon\text{-closed}}(X) \equiv \forall X' \in \times_{j=1..n} 2^{\mathcal{D}^j}, \left(X \sqsubseteq X' \wedge \mathcal{C}_{\epsilon\text{-connected}}(X')\right) \Rightarrow X = X'$.

Definition 7 **(Closed ET-n-set)** Given an n-set X, X is a closed ET-n-set iff $\mathcal{C}_{\epsilon\text{-connected}}(X) \wedge \mathcal{C}_{\epsilon\text{-closed}}(X)$.

Let us discuss the meaning of the noise-tolerance parameters on a closed ET-n-set $(X^1, \ldots, X^n)$. The parameter ϵ^i quantifies, on any element in X^i, the maximal number of n-tuples that are allowed to be absent from $\mathcal{R}$. In other terms, with a spatial vision of a pattern (an n-dimensional rectangle in $\mathcal{R}$ modulo permutations of the hyperplanes), ϵ^i is the maximal number of '0's on any hyperplane of the ith dimension. Furthermore $\mathcal{C}_{\epsilon\text{-closed}}$ forces $(X^1, \ldots, X^n)$ to be closed, i. e., any extension of it violates $\mathcal{C}_{\epsilon\text{-connected}}$. If $\forall i = 1..n, \epsilon^i = 0$ then $\mathcal{C}_{\epsilon\text{-connected}} \equiv \mathcal{C}_{\text{connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}} \equiv \mathcal{C}_{\text{closed}}$. The closed ET-$n$-set is a generalization of the closed n-set.

It is said that the definition of a closed ET-n-set uses an absolute noise-tolerance because it considers numbers rather than proportions of '0's. The following function helps in referring to counts of n-tuples absent from $\mathcal{R}$ on any element of an n-set.

Definition 8 **(Function 0)** Given an n-set X, $\forall i = 1..n, \forall e \in \mathcal{D}^i, 0(X, e) = |(X^1 \times \cdots \times \{e\} \times \cdots \times X^n) \setminus \mathcal{R}|$.

Example 2 Table 2 shows the hyperplane of the 3-set $X = (\{\alpha, \gamma\}, \{1, 2, 3\}, \{B\})$ related to the element B in the third attribute domain of $\mathcal{R}_E$ (Table 1). It contains only one absent 3-tuple, $(\gamma, 3, B)$, i. e., $0(X, B) = 1$.

The following property states that the function $0(., e)$ is increasing w.r.t. $\sqsubseteq$ and its proof is straightforward.

Property 1 *($0(., e)$ is increasing)* $\forall i = 1..n, \forall e \in \mathcal{D}^i, \forall (X, X') \in (\times_{j=1..n} 2^{\mathcal{D}^j})^2$,

$$X \sqsubseteq X' \Rightarrow 0(X, e) \leq 0(X', e).$$

Table 2 Hyperplane of $(\{\alpha, \gamma\}, \{1, 2, 3\}, \{B\})$ related to the element B

1	1	1
2	1	1
3	1	0
	α	γ

Let us use the 0 function to rewrite Definitions 5 and 6:

Definition 9 ($\mathcal{C}_{\epsilon\text{-connected}}$) Given an n-set X, $\mathcal{C}_{\epsilon\text{-connected}}(X) \equiv \forall i = 1..n, \forall e \in X^i$, $0(X, e) \leq \epsilon^i$.

Definition 10 ($\mathcal{C}_{\epsilon\text{-closed}}$) Given an n-set X, $\mathcal{C}_{\epsilon\text{-closed}}(X) \equiv \forall i = 1..n, \forall e \in \mathcal{D}^i \setminus X^i$,
$$\begin{cases} 0(X, e) > \epsilon^i \\ \text{or} \\ \exists j \neq i, \exists f \in X^j \text{ s.t. } 0((X^1, \ldots, X^i \cup \{e\}, \ldots, X^n), f) > \epsilon^j \end{cases}.$$

Definition 9 is a direct rewriting of Definition 5. Definition 10 is more than that. It is equivalent to Definition 6 because if a pattern can be extended without violating $\mathcal{C}_{\epsilon\text{-connected}}$, then there is such an extension with *one* element only (and the reverse obviously is true too). Furthermore, Definition 10 details the two ways to break $\mathcal{C}_{\epsilon\text{-connected}}$. Either the element to extend the closed ET-n-set gathers, when projected on the pattern, too many n-tuples absent from $\mathcal{R}$ or this additional element makes the number of '0's on an orthogonal element (an element from another domain of the relation) exceed the related noise-tolerance parameter.

Example 3 Let $\epsilon = (1, 1, 1)$. $X = (\{\alpha, \gamma\}, \{1, 2, 3\}, \{B\})$ is a closed ET-3-set in $\mathcal{R}_E$. X satisfies $\mathcal{C}_{\epsilon\text{-connected}}$ since each of its hyperplanes contains, at most, one 3-tuple absent from $\mathcal{R}_E$: $0(X, \alpha) = 0, 0(X, \gamma) = 1, 0(X, 1) = 0, 0(X, 2) = 0, 0(X, 3) = 1$ and $0(X, B) = 1$. X satisfies $\mathcal{C}_{\epsilon\text{-closed}}$ because extending it with any additional element either means that the hyperplane of X on this element contains strictly more than one 3-tuple absent from $\mathcal{R}_E$ (e. g., $0(X, \beta) = 2$) *or* at least one of the hyperplanes on an orthogonal element in X would contain strictly more than one 3-tuple absent from $\mathcal{R}_E$ (e. g., 4 cannot extend X because $0((\{\alpha, \gamma\}, \{1, 2, 3, 4\}, \{B\}), B) = 2$). $(\alpha, \beta, \gamma\}, \{1, 2\}, \{A, B\})$ is another closed ET-3-set in $\mathcal{R}_E$.

If a pattern is *not* ϵ-connected then none of its super-patterns (w.r.t. $\sqsubseteq$) is:

Property 2 (Anti-monotonicity of $\mathcal{C}_{\epsilon\text{-connected}}$) $\forall (X, X') \in (\times_{j=1..n} 2^{\mathcal{D}^j})^2$,

$$\left(X \sqsubseteq X' \wedge \neg\mathcal{C}_{\epsilon\text{-connected}}(X)\right) \Rightarrow \neg\mathcal{C}_{\epsilon\text{-connected}}(X').$$

Proof By Definition 9, $\neg\mathcal{C}_{\epsilon\text{-connected}}(X)$ means that $\exists i = 1..n, \exists e \in X^i$ s.t. $0(X, e) > \epsilon^i$. By Definition 2, $(e \in X^i \wedge X \sqsubseteq X') \Rightarrow e \in X'^i$. Finally, by Property 1, $0(X', e) \geq 0(X, e) > \epsilon^i$, i. e., $\neg\mathcal{C}_{\epsilon\text{-connected}}(X')$. □

If a pattern is *not* ϵ-closed then none of its sub-patterns (w.r.t. $\sqsubseteq$) is:

Property 3 (Monotonicity of $\mathcal{C}_{\epsilon\text{-closed}}$) $\forall (X, X') \in (\times_{j=1..n} 2^{\mathcal{D}^j})^2$,

$$\left(X \sqsubseteq X' \wedge \neg\mathcal{C}_{\epsilon\text{-closed}}(X')\right) \Rightarrow \neg\mathcal{C}_{\epsilon\text{-closed}}(X).$$

Proof By Definition 10, $\neg\mathcal{C}_{\epsilon\text{-closed}}(X')$ means that $\exists i = 1..n, \exists e \in \mathcal{D}^i \setminus X'^i$ such that $0(X', e) \leq \epsilon^i \wedge \forall j \neq i, \forall f \in X'^j, 0((X'^1, \ldots, X'^i \cup \{e\}, \ldots, X'^n), f) \leq \epsilon^j$. By Definition 2, $(e \in \mathcal{D}^i \setminus X'^i \wedge X \sqsubseteq X') \Rightarrow e \in \mathcal{D}^i \setminus X^i$. By Property 1, $0(X, e) \leq 0(X', e) \leq \epsilon^i$ (1). By Definition 2, $\forall j \neq i, \forall f \in X'^j$, $(f \in X^j \wedge X \sqsubseteq X') \Rightarrow f \in X'^j$ and, by Property 1, $0((X^1, \ldots, X^i \cup \{e\}, \ldots, X^n), f) \leq 0((X'^1, \ldots, X'^i \cup \{e\}, \ldots, X'^n), f) \leq \epsilon^j$ (2). Together, (1) and (2) mean $\neg\mathcal{C}_{\epsilon\text{-closed}}(X)$. □

2.4 Relative noise-tolerance

Intuitively, one can imagine that taking into account the sizes of the various dimensions of a pattern should help to identify more relevant error-tolerant ones. Yet our definition of a closed ET-n-set is based on absolute parameters and not relative ones. This section will actually conclude on the significant issues raised by a relative tolerance to noise, hence our choice of an absolute one. Nevertheless, let us first show that our closed ET-n-set extractor, named Fenster, can be adapted to enumerate patterns with a relative tolerance to noise. This adaptation restricts the pattern search space to a region where a chosen relative tolerance to noise can be converted into an absolute one. This restriction is possible and efficient because Fenster can enforce any *piecewise (anti)-monotone constraint* along the extraction (search space pruning). A constraint simply is a predicate taking in argument an n-set, i. e., every dimension of a pattern. That constraint complements Definition 7, i. e., it defines additional properties the closed ET-n-sets must have. A constraint is piecewise (anti)-monotone if fixing all occurrences of all variables but one (occurrence) always provides a constraint that is monotone (if the constraint is satisfied and the free occurrence grows, w.r.t. $\subseteq$, then it keeps on being satisfied) or anti-monotone (if the constraint is violated and the free occurrence grows, w.r.t. $\subseteq$, then it keeps on being violated) Cerf et al. (2009a).

Many popular constraints deal with the "geometry" of the patterns, i. e., disregard the actual elements in it and only consider the cardinalities of its dimensions: the famous minimal size constraint forces one of the dimension to contain at least $\mu \in \mathbb{N}$ elements (where μ is a parameter fixed by the analyst), the minimal area constraint forces the product of the dimension sizes to exceed a threshold $\nu \in \mathbb{N}$, etc. An n-dimensional Cartesian coordinate system allows to depict the pattern selection imposed by such a constraint. The coordinates of a pattern are its numbers of elements in each attribute domain of the n-ary relation. For instance, a pattern $(X^1, X^2) \in 2^{\mathcal{D}^1} \times 2^{\mathcal{D}^2}$ in a binary relation $\mathcal{R} \subseteq \mathcal{D}^1 \times \mathcal{D}^2$ is represented by the point $(|X^1|, |X^2|)$ in the Cartesian plane. The patterns satisfying a minimal area constraint relate to points above the curve whose equation is $xy = \nu$ ($\nu \in \mathbb{N}$ being the minimal area chosen by the analyst).

A specific piecewise (anti)-monotone constraint allows to force the "geometry" of a pattern to be in a region of the n-dimensional Cartesian coordinate system where its absolute tolerance to noise corresponds to a relative one. In other words, instead of an absolute tolerance to noise, the analyst can specify a relative one $(r^i)_{i=1..n} \in [0, 1]^n$

$\lvert X^2\rvert \in \left[\frac{2}{r^1}, \frac{3}{r^1}\right[$	$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$\lvert X^2\rvert \in \left[\frac{1}{r^1}, \frac{2}{r^1}\right[$	$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$\lvert X^2\rvert \in \left[0, \frac{1}{r^1}\right[$	$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
	$\lvert X^1\rvert \in \left[0, \frac{1}{r^2}\right[$	$\lvert X^1\rvert \in \left[\frac{1}{r^2}, \frac{2}{r^2}\right[$	$\lvert X^1\rvert \in \left[\frac{2}{r^2}, \frac{3}{r^2}\right[$	$\lvert X^1\rvert \in \left[\frac{3}{r^2}, \frac{4}{r^2}\right[$

Fig. 1 Conversion from a relative noise tolerance (r^1, r^2) to an absolute one (ϵ^1, ϵ^2) depending on the region of interest

and a region of interest. The relative parametrization is converted into the corresponding absolute one (parameters $(\epsilon^i)_{i=1..n} \in \mathbb{N}^n$) and FENSTER extracts the desired closed ET-n-sets in the chosen region. Here is the constraint defining such a region:

$$\mathcal{C}_{\text{in-region-of-interest}}(X) \equiv \bigwedge_{i=1}^{n} \left(\epsilon^i \leq r^i \prod_{j \neq i} |X^j| < \epsilon^i + 1 \right).$$

The proof of the piecewise (anti)-monotonicity of $\mathcal{C}_{\text{in-region-of-interest}}$ is based on, first, splitting the double inequalities, then, showing that the left ones are anti-monotone and the right ones monotone.

The regions of interest derive from the relative parameters $(r^i)_{i=1..n}$ and it is easy to compute the absolute parameters $(\epsilon^i)_{i=1..n}$ matching, in a given region, the desired patterns. Figure 1 depicts these regions in the case of a binary relation, i. e., it plots their contour lines in the Cartesian plane whose coordinate system was described earlier. The couple of integers inside a region are the absolute parameters FENSTER uses. When $n \geq 3$, analog regions can be drawn in the n-dimensional coordinate system $(|X^1|, \ldots, |X^n|)$ but they are not rectangular anymore.

If the analyst is interested in (relatively defined) closed ET-n-sets scattered across several regions, FENSTER has to be run several times. Each run is performed under the constraint $\mathcal{C}_{\text{in-region-of-interest}}$ corresponding to the region and with the associated absolute tolerance to noise. The union of the closed ET-n-sets listed at each run is the collection the analyst queried. Because the regions do not overlap, this union actually is a concatenation, i. e., every desired pattern is discovered only once. Nevertheless, there may be patterns that are included into each other (according to $\sqsubset$). This is expected with a relative tolerance to noise: there may exist a small pattern tolerating about the *same proportion* of noise as a larger one. The larger one would encompass a *greater absolute* quantity of noise on at least one of its elements. Only keeping the largest patterns (i. e., those that are not including in another output pattern) does not provide a lossless condensation of all patterns. In other terms, given an n-set tolerating proportions of noise, some sub-patterns of it (according to $\sqsubset$) gather greater proportions of noise. That is why the closed ET-itemset miners, tolerating proportions of noise, do not enforce a closedness constraint and extract huge collections of redundant patterns. To the best of our knowledge, the only exception to this rule is `AC-Close` (Cheng et al. 2008) and the closed patterns it outputs are not a lossless condensation of all (closed and unclosed) patterns.

By complementing Definition 7 with $\mathcal{C}_{\text{in-region-of-interest}}$, FENSTER discovers patterns that are closed in the region. However, this closedness is not lossless either. To avoid this issue, *all* ET-n-sets, tolerating noise in a relative way, could be enumerated. In other terms, $\mathcal{C}_{\text{in-region-of-interest}}$ would only complement Definition 9. As a consequence, the algorithm detailed in the next section would be simpler (no need of for the $\mathcal{S}$ n-set). Its execution would require far more time though. Indeed, no closedness constraint means far more patterns to list. Anyway, with or without the closedness constraint, a relative tolerance to noise entails significantly increased time requirements (in comparison with an absolute tolerance to noise). Indeed, it offers far less opportunities of search space pruning. When defined in an absolute way (as in Definition 9), $\mathcal{C}_{\epsilon\text{-connected}}$ is anti-monotone, i. e., it allows the extractor to ignore the super-patterns of any n-set exceeding the authorized ϵ^i absent n-tuples per element. Indeed, the violated absolute thresholds are violated as well by the larger patterns. In the same situation (enumeration of a pattern violating the ϵ-connectedness) but with relative noise-tolerance thresholds, the extractor often has to enumerate these super-patterns. Like the state of the Art, it takes FENSTER far more time to list patterns tolerating proportions of noise in a large number of regions of interest (hence, many extractions) than patterns tolerating a comparable amount of noise but in an absolute way.

Both the absence of a lossless condensation of the patterns (hence, much redundant patterns) and the absence of an anti-monotone connectedness constraint (hence, the practical impossibility to mine large datasets) justify our choice to mine n-sets tolerating absolute quantities of noise.

3 Introducing the FENSTER algorithm

3.1 Search space recursive partitioning

Like many complete algorithms for constraint-based local pattern mining (including Cerf et al. (2009a), the state-of-the-art algorithm for n-ary relation mining), FENSTER is based on enumerating candidates in a way that can be represented by a binary tree where (a) at every node, an element e is enumerated; (b) every pattern extracted from the left child *does contain* e; (c) every pattern extracted from the right child does *not* contain e. This provides a partition of the search space, i. e., the union of the closed ET-n-sets found in both enumeration sub-trees are exactly the closed ET-n-sets to be extracted from the parent node (*correctness*) *and* each of these closed n-sets is found only once (*unicity*). In the case of FENSTER, the enumerated element e can always be freely chosen among all the elements (from all attribute domains $\mathcal{D}^1$, $\mathcal{D}^2$, …, $\mathcal{D}^n$) remaining in the search space.

Each node N in the enumeration tree is a pair (U, V) where U and V are two n-sets. N *represents* all the n-sets containing every element in U and a subset of the elements in V. In other words, N defines the search space region that contains every n-set X s.t. $U \sqsubseteq X \sqsubseteq U \sqcup V$. The root node, $((\emptyset, \ldots, \emptyset), (\mathcal{D}^1, \ldots, \mathcal{D}^n))$, represents all possible n-sets. On the contrary, any node of the form $(U, (\emptyset, \ldots, \emptyset))$ represents a single n-set: U. More generally, a node (U, V) represents $2^{\sum_{i=1}^{n} |V^i|}$ n-sets.

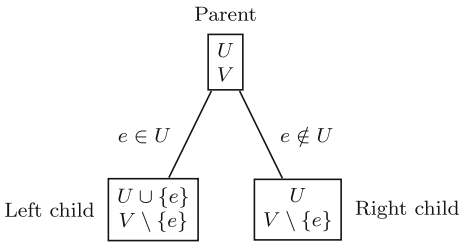


Fig. 2 Enumeration of any element $e \in V$

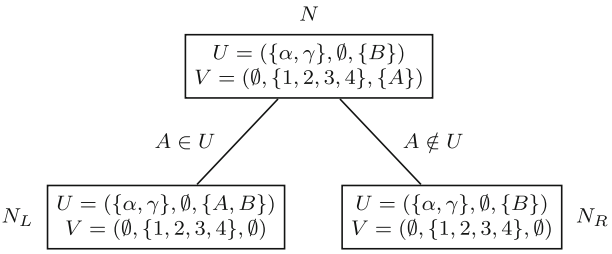


Fig. 3 Enumeration of the element $A \in V^3$ from node N (Example 5)

Example 4 The enumeration node $N = (U, V)$ where $U = (\{\alpha, \gamma\}, \emptyset, \{B\})$ and $V = (\emptyset, \{1, 2, 3, 4\}, \{A\})$ represents 2^5 (i. e., 32) 3-sets. E. g., it represents $(\{\alpha, \gamma\}, \emptyset, \{B\})$ and $(\{\alpha, \gamma\}, \{1, 2\}, \{B\})$. On the contrary, it represents neither $(\{\alpha, \gamma\}, \{1, 2\}, \emptyset)$ (B *must* be in the 3-set) nor $(\{\alpha, \beta, \gamma\}, \{1, 2\}, \{B\})$ (β *must not* be in the 3-set).

At a node $N = (U, V)$, FENSTER chooses an element e from V (the selection criterion is discussed in Sect. 4.2) and generates two new nodes, $N_L = (U_L, V_L) = (U \cup \{e\}, V \setminus \{e\})$ and $N_R = (U_R, V_R) = (U, V \setminus \{e\})$. N_L (resp. N_R) represents the n-sets of N that contain (resp. do not contain) e. Figure 2 depicts this simple partitioning of the search space.

Example 5 Considering the node N of Example 4, the selection of the element $A \in V^3$ leads to the two nodes $N_L = ((\{\alpha, \gamma\}, \emptyset, \{A, B\}), (\emptyset, \{1, 2, 3, 4\}, \emptyset))$ and $N_R = ((\{\alpha, \gamma\}, \emptyset, \{B\}), (\emptyset, \{1, 2, 3, 4\}, \emptyset))$ (see Fig. 3).

3.2 Efficient enforcement of $\mathcal{C}_{\epsilon\text{-connected}}$

Property 2 allows to reduce the search space of the left child, i. e., the number of elements in V_L. Indeed, by Property 2, if an element $v \in V_L$ is such that $\neg\mathcal{C}_{\epsilon\text{-connected}}(U_L \cup \{v\})$, then $\forall X \supseteq U_L \cup \{v\}$, $\neg\mathcal{C}_{\epsilon\text{-connected}}(X)$. In other terms, among the n-sets represented by (U_L, V_L), those involving such an element v necessarily violate $\mathcal{C}_{\epsilon\text{-connected}}$. As a consequence, all such elements (i. e., $\{v \in V_L \mid \neg\mathcal{C}_{\epsilon\text{-connected}}(U_L \cup \{v\})\}$) are safely removed from the search space V_L. Figure 4 depicts this process, which also guarantees that, at any node (U, V) of the enumeration tree, U can be "augmented" with any element $e \in V$ and necessarily remains ϵ-connected. In this way, and ignoring the enforcement of $\mathcal{C}_{\epsilon\text{-closed}}$ detailed in the next section, every n-set satisfying

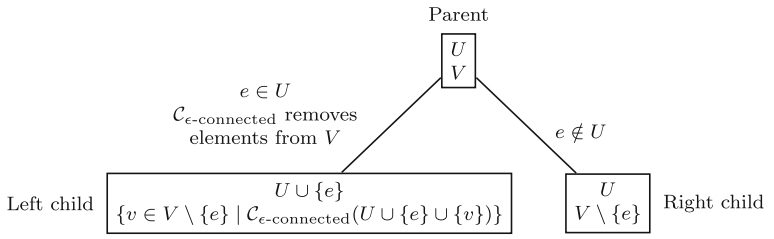


Fig. 4 Enumeration of any element $e \in V$. $\mathcal{C}_{\epsilon\text{-connected}}$ removes elements from V

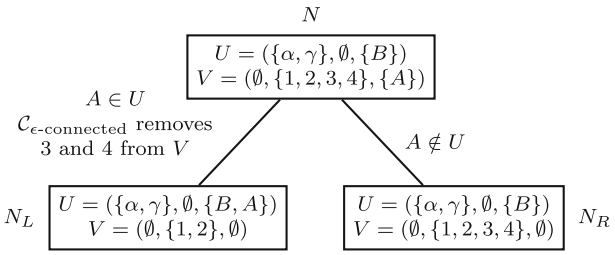


Fig. 5 Illustration of Example 6

$\mathcal{C}_{\epsilon\text{-connected}}$ is considered once (and only once) along the enumeration. The performance costs pertaining to the verification of $\mathcal{C}_{\epsilon\text{-connected}}$ are fundamental and are detailed in Sect. 4.1.

Example 6 Let $\epsilon = (1, 1, 1)$. In our running example, and according to Table 1, neither the element 3 nor the element 4 can be added to $U_L = (\{\alpha, \gamma\}, \emptyset, \{A, B\})$ to form a 3-set satisfying $\mathcal{C}_{\epsilon\text{-connected}}$. Indeed, neither $\mathcal{C}_{\epsilon\text{-connected}}(\{\alpha, \gamma\}, \{3\}, \{B\})$ nor $\mathcal{C}_{\epsilon\text{-connected}}(\{\alpha, \gamma\}, \{4\}, \{B\})$ is true (see Fig. 5).

Until now, we discussed how to extract all n-sets satisfying $\mathcal{C}_{\epsilon\text{-connected}}$ in n-ary relations. We now need to enforce the closedness property.

3.3 Efficient enforcement of $\mathcal{C}_{\epsilon\text{-closed}}$

For better performances, the closedness constraint must be handled during the enumeration process (safe pruning) rather than in a post-processing phase. Property 3 supports this achievement. At any enumeration node $N = (U, V)$, if there exists an element $s \in (\cup_{i=1..n}\mathcal{D}^i) \setminus (\cup_{i=1..n}U^i \cup V^i)$ such that $\mathcal{C}_{\epsilon\text{-connected}}(U \sqcup V \cup \{s\})$ is satisfied, then, by Definition 6, $U \sqcup V$ is not ϵ-closed and, by Property 3, neither is an n-set $X \sqsubseteq U \sqcup V$. Because N represents n-sets that are all sub-patterns of $U \sqcup V$, the whole enumeration sub-tree rooted by N is safely pruned. FENSTER does not miss the closed ET-n-sets "containing" U (according to $\sqsubseteq$): they are found in the parts of the enumeration tree where $s \in U$.

One of the most interesting advantages of our enumeration strategy is that there is actually no need to check whether *every* element in $(\cup_{i=1..n}\mathcal{D}^i) \setminus (\cup_{i=1..n}U^i \cup V^i)$ may prevent $U \sqcup V$ from being ϵ-connected. Any element that has been removed from

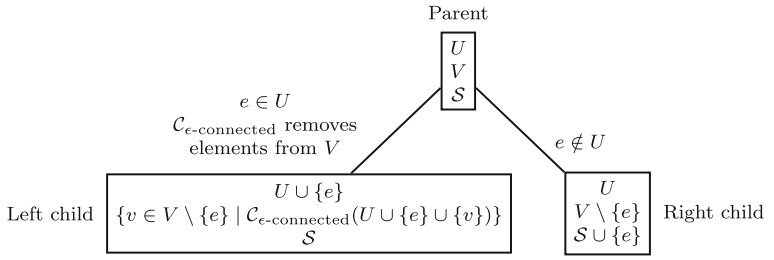


Fig. 6 Enumeration of any element $e \in V$. $\mathcal{C}_{\epsilon\text{-connected}}$ removes elements from V. $\mathcal{C}_{\epsilon\text{-closed}}$ is checked on $U \sqcup V$ extended with every element in $\mathcal{S}$

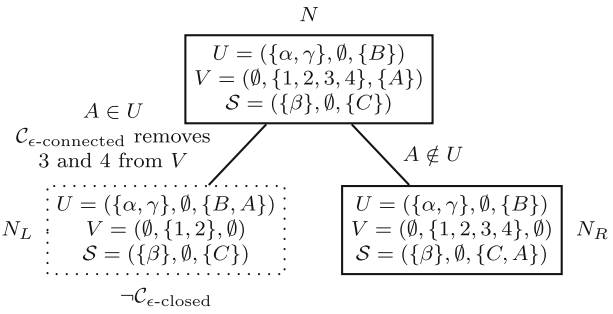


Fig. 7 Illustration of Example 7

V thanks to $\mathcal{C}_{\epsilon\text{-connected}}$ (see Sect. 3.2) cannot. Indeed, the reason of the removal of such an element f from V is that $\mathcal{C}_{\epsilon\text{-connected}}(U \cup \{f\})$ is false. In such circumstances, and whatever V, Property 2 guarantees that $\mathcal{C}_{\epsilon\text{-connected}}(U \sqcup V \cup \{f\})$ cannot be true either. When checking $\mathcal{C}_{\epsilon\text{-closed}}$ the only elements that need to be tried as extensions are those that were previously chosen to be enumerated but refused (right child). These elements are stored in an n-set that will always be denoted $\mathcal{S}$. Figure 6 complements Fig. 4 with this n-set $\mathcal{S}$.

Example 7 Still using the running example, where $\epsilon = (1, 1, 1)$, assume that $\mathcal{S} = (\{\beta\}, \emptyset, \{C\})$ is tied to the enumeration node N. At N_R, the enumerated element, A, is appended to its $\mathcal{S}$ 3-set. In this example, N_L violates $\mathcal{C}_{\epsilon\text{-closed}}$. Indeed, $\beta \in \mathcal{S}$ can extend $U \sqcup V$, i.e., $\mathcal{C}_{\epsilon\text{-connected}}(\{\alpha, \beta, \gamma\}, \{1, 2\}, \{B, A\})$ is true. $\mathcal{C}_{\epsilon\text{-closed}}$ prunes the enumeration sub-tree that N_L would root. In Fig. 7, illustrating this example, this node is framed with dots.

3.4 Two performance improvements

Every n-set X represented by a node $N = (U, V)$ is such that $U \sqsubseteq X$. By Property 2, if an element $s \in \mathcal{S}$ is such that $\neg\mathcal{C}_{\epsilon\text{-connected}}(U \cup \{s\})$, then $\forall X \sqsupseteq U \cup \{s\}$, $\neg\mathcal{C}_{\epsilon\text{-connected}}(X)$. As a consequence, s cannot prevent any n-set represented by N from being ϵ-closed. All such elements (i.e., $\{s \in \mathcal{S} \mid \neg\mathcal{C}_{\epsilon\text{-connected}}(U \cup \{s\})\}$) are safely removed from $\mathcal{S}$, whose only purpose precisely is the enforcement of $\mathcal{C}_{\epsilon\text{-closed}}$.

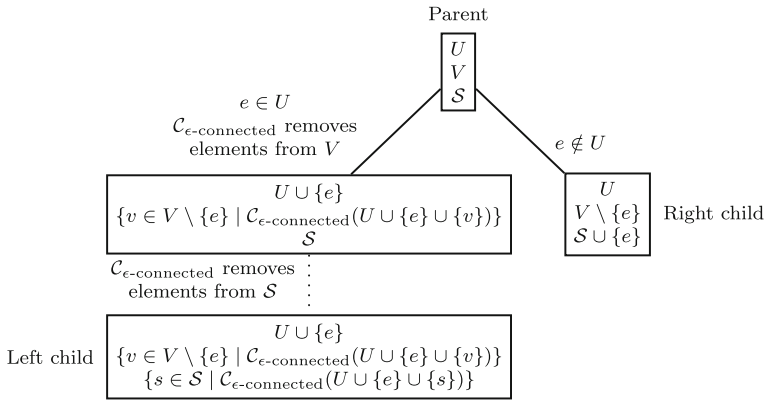


Fig. 8 FENSTER enumerating any element $e \in V$

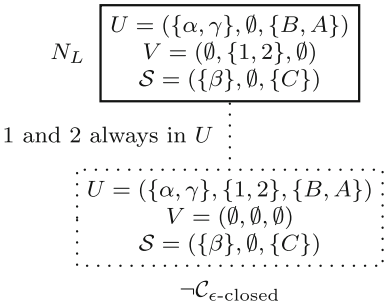


Fig. 9 Illustration of Example 8

The involved process is similar to the enforcement of $\mathcal{C}_{\epsilon\text{-connected}}$ (see Sect. 3.2) but applied on $\mathcal{S}$ instead of V (the fundamental implementation details are described in Sect. 4.1). This optimization speeds up the enforcement of $\mathcal{C}_{\epsilon\text{-closed}}$ for all the nodes deriving from N. This slight modification of enumeration is reported in Fig. 8.

In general, and contrary to the extraction of *exact* patterns, several closed ET-n-sets can be made by moving some elements from V to U. However, when $U \sqcup V$ satisfies $\mathcal{C}_{\epsilon\text{-connected}}$ then it is, by Definition 6, the only closed ET-n-set with U "included" (w.r.t. Definition 2). That is why $\mathcal{C}_{\epsilon\text{-connected}}(U \sqcup V)$ is tested. If it is satisfied, a direct jump to the leftmost leaf of the enumeration sub-tree (rooted by the current node) is performed. This jump is safe,

Because most of the nodes are at the bottom of the enumeration tree (in a complete binary tree, half of the nodes are leaves), this improvement significantly reduces the extraction times.

Example 8 In the enumeration node N_L of Example 6, $U \sqcup V = (\{\alpha, \gamma\}, \{1, 2\}, \{B, A\})$ and $\mathcal{C}_{\epsilon\text{-connected}}(\{\alpha, \gamma\}, \{1, 2\}, \{B, A\})$ is true. The improvement is applied (see Fig. 9) before the actual enforcement of $\mathcal{C}_{\epsilon\text{-closed}}$. Of course, the obtained node is not ϵ-closed for the same reason as the intermediary node (see Example 6): $\beta \in \mathcal{S}^1$ can extend it.

Input: $U, V, \mathcal{S}$
Output: Every closed ET-n-set containing every element in U, possibly some elements in V, and satisfying a piecewise (anti)-monotone constraint $\mathcal{C}$
if $\mathcal{C}_{\epsilon\text{-connected}}(U \sqcup V)$ **then**
  $U \leftarrow U \sqcup V$
  $V \leftarrow (\emptyset, \ldots, \emptyset)$
end if
if $\mathcal{C}$ may be satisfied by an n-set descending from this node $\wedge \mathcal{C}_{\epsilon\text{-closed}}(U \sqcup V)$ **then**
  if $V = (\emptyset, \ldots, \emptyset)$ **then**
    output(U)
  else
    Choose $e \in V$
    FENSTER$(U \cup \{e\},$
      $\{v \in V \setminus \{e\} \mid \mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{v\})\},$
      $\{s \in \mathcal{S} \mid \mathcal{C}_{\epsilon\text{-connected}}(U \cup \{e\} \cup \{s\})\})$
    FENSTER$(U, V \setminus \{e\}, \mathcal{S} \cup \{e\})$
  end if
end if

Fig. 10 The FENSTER algorithm

3.5 Algorithm

FENSTER is a depth-first search algorithm. It takes three arguments: U, V and $\mathcal{S}$. It starts with $U_0 = (\emptyset, \ldots, \emptyset)$, $V_0 = (\mathcal{D}^1, \ldots, \mathcal{D}^n)$ and $\mathcal{S}_0 = (\emptyset, \ldots, \emptyset)$. Its major steps are presented in the pseudo-code of Fig. 10, which can be seen as a translation of the diagram in Fig. 8. First of all, the closedness pro is checked (see Sect. 3.3). If it is satisfied and no element remains to be enumerated, the n-set U is output. Otherwise an element e of V is chosen (Sect. 4.2 discusses this step) and the search space is split between the n-sets that contain e and those that do not (see Sect. 3.1 and 3.2). Finally, FENSTER is recursively called on the two related enumeration nodes. Moreover, any user-defined relevancy constraint $\mathcal{C}$ can be additionally enforced as long as it is piecewise (anti)-monotone (Cerf et al. 2009a).

4 Implementation

4.1 Exploiting $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$

4.1.1 Performance issue

Although the enumeration of FENSTER is inspired by that of DATA- PEELER (Cerf et al. 2009a), FENSTER is not a trivial extension of DATA- PEELER. A naive enforcement of the new constraints $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ would lead to disastrous running times. Contrary to DATA- PEELER, FENSTER cannot traverse small subspaces of the dataset in search of *one* n-tuple absent from $\mathcal{R}$. When an element e is chosen, the absence in $\mathcal{R}$ of *one* n-tuple with e (i. e., on the hyperplane related to e) is not enough to enforce $\mathcal{C}_{\epsilon\text{-connected}}$, whereas it is when enforcing $\mathcal{C}_{\text{connected}}$. Searching for *several* n-tuples absent from $\mathcal{R}$ in this hyperplane is not enough either. FENSTER needs to know the other n-tuples absent from $\mathcal{R}$ that were previously tolerated in every n-set represented by the current node, i. e., the n-tuples in $(\times_{i=1..n} U^i) \setminus \mathcal{R}$. It needs to know *where*,

i. e., on which hyperplanes, they are and how many of them are found on each of these hyperplanes. The enforcement of $\mathcal{C}_{\epsilon\text{-closed}}$ raises the same trouble: given U, V and $\mathcal{S}$, the ϵ-closedness of some n-set represented by (U, V) cannot be proved by only consulting with the n-tuples in $\times_{i=1..n} U^i \cup V^i$ involving the elements in $\mathcal{S}$. As a consequence, a naive enforcement of $\mathcal{C}_{\epsilon\text{-connected}}$ (resp. $\mathcal{C}_{\epsilon\text{-closed}}$) would, at every iteration, count the numbers of n-tuples absent from $\mathcal{R}$ in every hyperplane of U (resp. $U \sqcup V$) and on each of its projections on the elements in V (resp. $\mathcal{S}$). Such an implementation would be intractable even on rather small relations.

4.1.2 Noise counters in relevant subspaces of the relation

To remain tractable on large relations, FENSTER relies on the following observation: from a parent enumeration node to its children, U and $U \sqcup V$ do not change much. U only grows by one element in the left child and is left unchanged in the right child. $U \sqcup V$ loses one element in the right child; potentially more in the left child. To avoid repetitive scans of the same parts of $\mathcal{R}$, FENSTER updates counters of absent n-tuples that are relevant to the verification of Definitions 9 and 10 on the recursively considered patterns. In this way, at every recursive call, FENSTER only needs to access the parts of $\mathcal{R}$ related to the symmetric differences between the n-sets U_P and V_P of the parent node and the respective n-sets U and V of the child node. This means much better time performances (than the naive approach) to the cost of a worse memory consumption (to store the counters). Later, it will be formally shown that the time gain is huge while the space complexity usually is dominated by the dataset when $n \geq 4$.

Let us finally list the counters that are relevant when enforcing $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$. Keeping in mind their definitions (i. e., Definitions 9 and 10) while looking at Fig. 10 provides the following list:

To check $\mathcal{C}_{\epsilon\text{-closed}}(U \sqcup V)$:
- $\forall s \in \mathcal{S}, 0(U \sqcup V, s)$;
- $\forall s \in \mathcal{S}, \forall u \in U, 0(U \sqcup V \cup \{s\}, u)$.

Given $e \in V$ and $U_L = U \cup \{e\}$ (the elements that are present in every n-set descendant of the left child), to compute $\{v \in V \setminus \{e\} \mid \mathcal{C}_{\epsilon\text{-connected}}(U_L \cup \{v\})\}$:
- $\forall v \in V, 0(U_L, v)$;
- $\forall v \in V, \forall u \in U_L, 0(U_L \cup \{v\}, u)$.

Given $e \in V$ and $U_L = U \cup \{e\}$ (the elements that are present in every n-set descendant of the left child), to compute $\{s \in \mathcal{S} \mid \mathcal{C}_{\epsilon\text{-connected}}(U_L \cup \{s\})\}$:
- $\forall s \in \mathcal{S}, 0(U_L, s)$;
- $\forall s \in \mathcal{S}, \forall u \in U_L, 0(U_L \cup \{s\}, u)$.

By factorizing the last two points, four families of counters are useful:

- $\forall f \in \mathcal{S}, 0(U \sqcup V, f)$;
- $\forall f \in \mathcal{S}, \forall u \in U, 0(U \sqcup V \cup \{f\}, u)$;
- $\forall f \in V \sqcup \mathcal{S}, 0(U_L, f)$;
- $\forall f \in V \sqcup \mathcal{S}, \forall u \in U_L, 0(U_L \cup \{f\}, u)$.

During the recursion, any element in V may, in the descendant nodes, belong to a U or a $\mathcal{S}$ n-set. By keeping updated every type of counter ($0(U \sqcup V, f), 0(U \sqcup V \cup \{f\}, u)$,

$0(U_L, f)$ and $0(U_L \cup \{f\}, u)$) for every $(f, u) \in (U \sqcup V \sqcup \mathcal{S})^2 \setminus U^2$, their computation is always performed incrementally. In this way, some counters are only used when the element defining the hyperplane is in a specific set (e. g., a counter $0(U \sqcup V, f)$ is not used until $f \in \mathcal{S}$). Anyway, it is advantageous to maintain them updated for every element that may reach a state where they would be useful (e. g., $0(U \sqcup V, f)$ is maintained updated even if $f \in V$). An alternative strategy would be to initialize a counter when required. It would be less efficient because, along the enumeration tree, there are exponentially many states where a given counter is useful. As a consequence, the cost of an on-demand initialization of the counter (scan of part of the dataset) multiplied by this number of states exceeds the cost of maintaining them all updated until used or useless. Thus, all counters are initialized while storing the dataset and, whenever elements are moved or removed from V, the counters are updated by only traversing the symmetric differences between the n-sets U_P and V_P of the parent node and the respective n-sets U and V of a child node. These updates can be further improved for the counters of the types $0(U_L \cup \{f\}, u)$ and $0(U \sqcup V \cup \{f\}, u)$ (where $(f, u) \in (U \sqcup V \sqcup \mathcal{S})^2 \setminus U^2$). To do so, they are replaced by:

- $0\,(U_L, f, u) = |\left(U_L^1 \times \cdots \times \{f\} \times \cdots \times \{u\} \times \cdots \times U_L^n\right) \setminus \mathcal{R}|$;
- $0\,(U \sqcup V, f, u) = |\left(U^1 \cup V^1 \times \cdots \times \{f\} \times \cdots \times \{u\} \times \cdots \times U^n \cup V^n\right) \setminus \mathcal{R}|$.

The desired quantities $0(U_L \cup \{f\}, u)$ and $0(U \sqcup V \cup \{f\}, u)$ can be computed, still without any access to the relation:

- $0\,(U_L \cup \{f\}, u) = 0\,(U_L, u) + 0\,(U_L, f, u)$;
- $0\,(U \sqcup V \cup \{f\}, u) = 0\,(U \sqcup V, u) + 0\,(U \sqcup V, f, u)$.

The two new types of counter involve much smaller subspaces (one dimension less) than the ones they replace. Thus, they do not need to be updated as often, hence the additional time gain.

Example 9 Consider that FENSTER, working on the relation $\mathcal{R}_E$, reaches the enumeration node where $U = (\{\alpha, \gamma\}, \{1, 2\}, \{B\})$ and $V = (\emptyset, \{3, 4\}, \emptyset)$. Consider, moreover, that $\mathcal{S} = (\{\beta\}, \emptyset, \{A\})$. Although this enumeration node is rather small, it is associated with too many counters to list them all here. Among them, $0(U, \alpha) = 0$, $0(U \sqcup V, \alpha) = 1$, $0(U, 3) = 1$, $0(U \sqcup V, 3) = 1$, $0(U, A) = 0$, $0(U \sqcup V, A) = 2$, $0(U, \alpha, 3) = 0, 0(U \sqcup V, \alpha, 3) = 0, 0(U, \alpha, A) = 0, 0(U \sqcup V, \alpha, A) = 2, 0(U, 3, A) = 1$, $0(U \sqcup V, 3, A) = 1$, etc.

4.1.3 Time gain

Consider an enumeration node (U, V), its n-set $\mathcal{S}$ and the enumerated element $e \in V$. With the naive enforcement of $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ (verifying Definitions 9 and 10 without the help of any counter), the n-tuples in $\cup_{i=1}^{n}(U^i \cup V^i \cup \mathcal{S}^i) \times (\times_{j \neq i} U^j \cup V^j)$ would be traversed at any child of (U, V). This set of n-tuples relates to the enforcement of $\mathcal{C}_{\epsilon\text{-closed}}$ only but it includes $\times_{i=1..n} U^i$ that is accessed to verify $\mathcal{C}_{\epsilon\text{-connected}}$ (at the left child only). Updating the counters, described in the previous section, requires accessing far less n-tuples. Assuming e was chosen in the dth dimension, FENSTER traverses $\cup_{i \neq d}(U^i \cup V^i \cup \mathcal{S}^i) \times (\times_{j \notin \{d,i\}} U^j \cup V^j)$ at any child of (U, V). The enumeration strategy, hence the number of enumeration nodes, being the same in both

cases, it could be written that FENSTER is as fast as the naive approach running on a dataset with one dimension less (and the same number of elements per dimension).

However, this statement only holds when the time spent using the counters (to enforce $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$) does not dominate the time spent accessing the n-tuples (to update the counters). To verify it, the number of counters accessed at every node must be studied. To enforce $\mathcal{C}_{\epsilon\text{-connected}}$, it is, at worst (none of the elements in V are removed), $|V| + 2\sum_{i=1..n} |V^i| \sum_{j\neq i} |U^j|$. To enforce $\mathcal{C}_{\epsilon\text{-closed}}$, it is, at worst (none of the elements in $\mathcal{S}$ extends $U \sqcup V$), $|\mathcal{S}| + 2\sum_{i=1..n} |\mathcal{S}^i| \sum_{j\neq i} |U^j \cup V^j|$. In both cases, the first term relates to checking whether every hyperplane $v \in V$ (resp. $s \in \mathcal{S}$) contains too many (resp. enough) n-tuples absent from $\mathcal{R}$ to satisfy $\mathcal{C}_{\epsilon\text{-connected}}$ (resp. $\mathcal{C}_{\epsilon\text{-closed}}$) and the second term relates to checking whether an hyperplane $v \in V$ (resp. $s \in \mathcal{S}$), if added to U (resp. $U \sqcup V$) would make any orthogonal element exceed its noise tolerance threshold. As explained at the end of the previous section, two counters are summed to obtain the number of absent n-tuples on an orthogonal hyperplane, hence the factor. Overall, the time spent using the counters is $O(\sum_{i=1..n} |\mathcal{S}^i| \sum_{j\neq i} |U^j \cup V^j|)$.

This number is now compared to the time spent updating the counters, i. e., $O(|\cup_{i\neq d} (U^i \cup V^i \cup \mathcal{S}^i) \times (\times_{j\notin\{d,i\}} U^j \cup V^j)|)$, which can be rewritten $O(\sum_{i\neq d} |U^i \cup V^i \cup \mathcal{S}^i| \prod_{j\notin\{d,i\}} |U^j \cup V^j|)$. When $n = 2$, the time spent accessing the counters, $O(|(\{\mathcal{S}^1\} \times (U^2 \cup V^2)) \cup (\{\mathcal{S}^2\} \times (U^1 \cup V^1))|)$, is dominant. It is, however, below the time required by the naive enforcement of $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$. When $n \geq 3$, the comparison depends on the actual cardinalities of the sets U^i, V^i and $\mathcal{S}^i$ ($i = 1..n$). In the worst case scenario we consider, $\mathcal{C}_{\text{connected}}$ never removes any element from V. As a consequence, every element that is not in $U \sqcup V$ was moved to $\mathcal{S}$ and, carrying on with our pessimistic view, the first improvement presented in Sect. 3.4 has not removed it from there. We therefore have $\forall i = 1..n$, $U^i \cup V^i \cup \mathcal{S}^i = \mathcal{D}^i$ and:

- $O\left(\sum_{i\neq d} |\mathcal{D}^i| \prod_{j\notin\{d,i\}} |\mathcal{D}^j \setminus \mathcal{S}^j|\right)$ accesses to n-tuples;
- $O\left(\sum_{i=1..n} |\mathcal{S}^i| \sum_{j\neq i} |\mathcal{D}^j \setminus \mathcal{S}^j|\right)$ accesses to counters.

For a clearer view of how large theses quantities are, let us assume that both the dataset and the pattern are perfectly cubic, i. e., $\exists (D, S) \in \mathbb{N}^2 \mid \forall i = 1..n, |\mathcal{D}^i| = D$ and $|\mathcal{S}^i| = S$. Under this assumption, we have:

- $O\left((n-1)D(D-S)^{n-2}\right)$ accesses to n-tuples;
- $O\left(nS(n-1)(D-S)\right)$ accesses to counters.

After a simplification by $(n-1)(D-S)$, we end up comparing $O(D(D-S)^{n-3})$ with $O(nS)$. When $n = 3$, the accesses to the counters and the n-tuples are on the same order, whereas the cost of traversing the relation dominates when $n \geq 4$. Here is a more intuitive way to understand it: at every enumeration node, the n-tuples that are accessed to update the counters are on an hyperplane (related to e in Fig. 10). It is an $(n-1)$-dimensional subspace. In contrast, the number of used counters, whatever the arity of $\mathcal{R}$, is on the order of a 2-dimensional subspace of the dataset (the finest counters are numbers of n-tuples absent from $\mathcal{R}$ at the intersection of *two* orthogonal hyperplanes). To conclude, it can indeed be written that, searching for closed patterns in an n-ary relation with $n \geq 3$, FENSTER is as fast as the naive approach (with no

counter) processing a relation on domains of the same sizes but with an arity of $n-1$. When $n=2$, a gain exists but is not as impressive.

4.2 Choosing the element to enumerate

At every recursive call, any element in V can be enumerated (function *Choose* in Fig. 10). Cerf et al. (2009a) empirically showed that the choice of this element, in particular the choice of the attribute domain it belongs to, is fundamental. Different sensible strategies produce different enumeration trees whose sizes (hence, the time required to traverse them) varies between several orders of magnitude. Compared to this previous work, FENSTER profits from more information at its disposal when it comes to choose an element to enumerate. The counters allow a finer choice and smaller enumeration trees are built. FENSTER chooses the enumerated element in two stages:

1. The attribute domain, in which the element will be enumerated, is chosen.
2. The element itself is chosen.

The chosen attribute domain $\mathcal{D}^d$ maximizes the following expression:

$$\sum_{k \neq d} \left(|V^k| \times \prod_{l \notin \{d,k\}} |U^l| \right).$$

This expression actually counts the n-tuples that are accessed to update the counters $0(U_L, f)$ and $0(U_L, f, u)$, where $f \in V^{k \neq d}$ and $u \in U^{l \notin \{d,k\}}$. These counters are those involved in the enforcement of $\mathcal{C}_{\epsilon\text{-connected}}$, i. e., in the removal of elements from $V \setminus \{e\}$ to V_L. As a consequence, the larger the expression above, the greater the expected reduction of the search space in the left child.

Then, FENSTER takes advantage of the counters $0(U_L, f)$. It chooses the element $f \in V^d$ providing the greatest $0(U_L, f)$. The justification is simple: the more n-tuples in $U^1 \times \cdots \times U^n$ that are absent from $\mathcal{R}$, the less room for others, hence the smaller the search space of the left child. For the same reason applied to the left grandchild (and beyond), when V^d maximizes $0(U, f)$, an element leading to a greater $0(U \sqcup V, f)$ is preferred.

Example 10 In the Example 6, illustrated by Fig. 5, the choice of enumerating $A \in V^3$ actually follows the heuristics stated above:

Choice of V^3: $\sum_{k \neq d} \left(|V^k| \times \prod_{l \notin \{d,k\}} |U^l| \right)$ is maximized for $d = 3$:
$d = 1$: $\left(|V^2| \times |U^3|\right) + \left(|V^3| \times |U^2|\right) = (4 \times 1) + (1 \times 0) = 4$;
$d = 2$: $\left(|V^1| \times |U^3|\right) + \left(|V^3| \times |U^1|\right) = (0 \times 1) + (1 \times 2) = 2$;
$d = 3$: $\left(|V^1| \times |U^2|\right) + \left(|V^2| \times |U^1|\right) = (0 \times 0) + (4 \times 2) = 8$.
Choice of A: Among the elements in V^3, $e = A$ maximizes $0((\{\alpha, \gamma\}, \emptyset, \{B\}), e)$ (this value is 0 but V^3 only contains A).

4.3 Space complexity

The n-sets U, V and $\mathcal{S}$ and all the counters associated with the elements they gather need to be copied whenever a left child enumeration node is built. When $n \in \{2, 3\}$, the counters occupy most of the memory. In this case, and since the depth of the recursion reaches, in the worst case, $\sum_{i=1}^{n} |D_i|$, the space complexity of FENSTER is $O((\sum_{i=1}^{n} |D_i|) \times (2 \sum_{i=1}^{n} |D_i| + 2 \sum_{i \neq j} |D_i \times D_j|)) = O(|D_i|^2 \times |D_j|)$, where D_i is the largest dimension and D_j the second largest. When $n \geq 4$, the space to store the relation, $O(\prod_{i=1}^{n} |D_i|)$, usually dominates that of the counters, which remains unchanged.

Because the second recursive call of FENSTER (construction of a right child) is a tail call, the variables (U, V, S and the counters) of the parent node can be safely reused. Not copying these variables reduces both the time and space requirements in a significant way. Overwriting the parent enumeration nodes with their left children (i. e., inverting the two recursive calls) would not provide as much gain. Indeed, in a right child enumeration node, the search space V is only reduced by one element and U stays unchanged. Because of that, the enumeration sub-tree rooted by a right child node is far less often pruned (by $\mathcal{C}_{\epsilon\text{-closed}}$ or $\mathcal{C}$) than that of a left child (where the search space V may be greatly reduced). As a consequence, the recursive calls of FENSTER (see Fig. 10), down to a leaf, usually involve far more right children than left ones and, in practical settings, overwriting the parent enumeration nodes with their right children significantly decreases the average number of nodes to be kept in memory. It even provides substantial gains in terms of average extraction time because the cost of copying all counters is taken off.

5 Empirical study on synthetic datasets

FENSTER is distributed under the terms of the GNU GPLv3[4]. It was coded in C++ and compiled with GCC 4.5.2. Most of the following experiments were performed on an Intel® processor cadenced at 2GHz, 2 GB of RAM and running a GNU/Linux™ operating system. Because `AC-Close` only runs on Windows™, the experiment involving it was performed on another computer equipped with an Intel® processor cadenced at 2.26GHz and 3 GB of RAM. The implementations of CUBEMINER (Ji et al. 2006), TRIAS (Jaschke et al. 2006), `AC-Close` (Cheng et al. 2008) and DCE (Georgii et al. 2011) were kindly provided by their respective authors. DCE's ability to rank the patterns was disabled.

5.1 Experimental setting

Four, possibly overlapping, n-sets are randomly placed in a *cubic* dataset, i. e., all attribute domains have the same cardinality. The obtained relation is named $\mathcal{R}_{\text{hidden}}$.

[4] http://dcc.ufmg.br/~lcerf/en/prototypes#fenster.

Some noise is added to it[5]. In this way, we obtain the relation $\mathcal{R}$ that is mined. The noise follows a Bernoulli distribution, i. e., every n-tuple has the same probability (called "noise level") to be switched (an n-tuple absent from $\mathcal{R}_{\text{hidden}}$ becomes present in $\mathcal{R}$ or vice versa). The experiments are performed with relations whose noise level varies between 0 and 0.45 (0.5 corresponds to purely random datasets). The mining task being symmetric w.r.t. the attributes, every tested parametrization satisfies $\forall i = 1..n, \epsilon^i = \epsilon \in \mathbb{N}$.

In this experimental section, FENSTER is compared to DCE and `AC-Close`. DCE (Georgii et al. 2011) is the only other existing approach able to mine noise tolerant-patterns in arbitrary n-ary relations. Contrary to FENSTER, its noise tolerance is *relative* (rather than *absolute*) and *per-pattern* (rather than *per-element*), i. e., the *proportion* of absent n-tuples covered by *a whole pattern* must exceed a density threshold for this pattern to be output by DCE. Since most of the extracted patterns barely satisfy the minimal size constraints, DCE's density thresholds are chosen so that they relate to the proportions of absent n-tuples FENSTER tolerates in these smallest patterns. For instance, considering a ternary relation, the extractions performed by FENSTER with $\epsilon = 2$ and at least four elements per attribute are compared with those performed by DCE with a minimal density of $1 - \frac{2}{4\times4} = 0.875$ and the same minimal size constraints.

Several approaches were designed to extract noise-tolerant patterns from *binary* relations (see Sect. 7.1). Three main reasons justify our choice of `AC-Close` as a competitor in this specific context (w.r.t. what FENSTER can achieve). First of all, its tolerance to noise is, like FENSTER's, *per-element*, i. e., per-transaction *and* per-item. Then, and again like FENSTER, `AC-Close` mines *closed* ET-patterns, whereas most of the other approaches do not force the returned itemsets to be closed. Finally, by constraining the cardinality of the *exact* support of a pattern to exceed αs (where s is a minimal size constraint on the ET-pattern and $\alpha \in [0, 1]$ is a user-defined parameter), `AC-Close` somehow circumvents the performance issues the other approaches go through when tolerating proportions of noise. Indeed, in their experimental section, the authors claim that `AC-Close` runs much faster than AFI, described in Liu et al. (2006).

As detailed in Sect. 7, the numerous other ET-itemset miners are less comparable to FENSTER. Notice, in particular, that the works of Ardian K. Poernomo and Vivekanand Gopalkrishnan allow the discovery of 1-dimensional patterns in binary relations, i. e., return subsets of items for which *there exists* at least one support so that the chosen definition of the pattern is verified. Because several 2-dimensional patterns may involve a same set of items, less patterns are listed and the tuples they cover are not directly accessible. This prevents the use of a post-process such as Cerf et al. (2009b), which heuristically agglomerates the patterns to tolerate more noise than what is possible, in a reasonable time, with complete approaches. Indeed, when listing *every* noise-tolerant pattern verifying the definition, the hidden patterns usually are unreachable in a reasonable time and heuristics are required to "finish the work". More precisely, the

[5] The commands generating the noisy relations and the Bash scripts we have run for these experiments are distributed under the terms of the GNU GPLv3. They are available at http://dcc.ufmg.br/~lcerf/en/utilities.html#noisy.

complete approaches can, in a real-life context, only list multiple fragments of the hidden patterns. If these fragments cover the same tuples as the hidden patterns (and only them), agglomerating the fragments allow to recover the hidden patterns. This explains our choice of a *tuple-based* measure to assess, in this section, the quality of the patterns extracted in a complete way:

Definition 11 (Quality) Given $\mathcal{E}$ the set of n-tuples covered by the extracted patterns and $\mathcal{R}_{\text{hidden}}$ the relation before the addition of noise, the quality of the extraction is $\frac{|\mathcal{R}_{\text{hidden}} \cap \mathcal{E}|}{|\mathcal{R}_{\text{hidden}} \cup \mathcal{E}|}$.

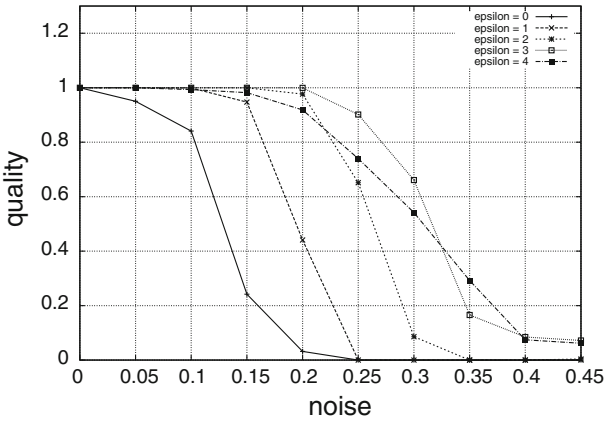
The other criteria used to evaluate the performances of FENSTER and its competitors are the number of patterns they list (the smaller, the faster the subsequent agglomeration) and the running time.

5.2 Quality results

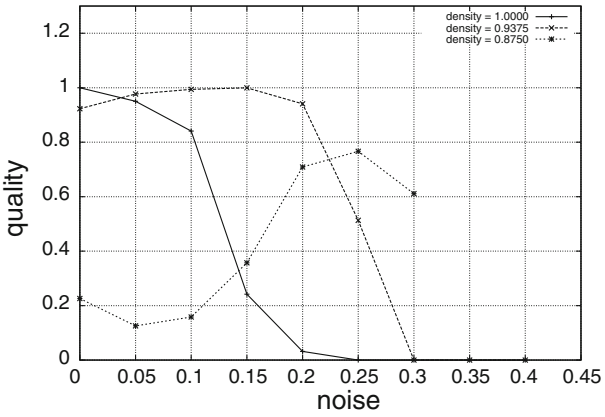
The qualities obtained with ternary relations are plotted in Fig. 11. In this setting, the hidden patterns contain eight elements in every attribute domain (of 32 elements) and the patterns are constrained to have at least four elements per attribute. According to Fig. 11a, the best parametrization for FENSTER is $\epsilon = 2$ when the level of noise is below 0.15. At this point, the quality of the extracted collection of closed ET-3-sets is almost perfect, whereas the collection of exact closed patterns shows a quality of 0.25. The noisiest settings are advantageously mined with $\epsilon = 3$. This confirms that greater noise tolerances are preferred to mine relations suffering from higher levels of noise.

Comparing these quality results with those of DCE allows to defend our choice of a per-element tolerance to noise. In Fig. 11b, DCE considers the noise covered by whole patterns and, doing so, the quality results are both worse and more sensitive to the choice of the noise tolerance threshold. In particular, tolerating a little more noise than necessary provides poor results. This is explained by the fact that an extracted pattern is somehow allowed to "go beyond" an hidden pattern if the covered part of the hidden pattern has not been altered by much noise. If this is the case, the tolerated proportion of absent tuples is far enough from being reached to accept additional elements outside the hidden pattern. Understanding this problem, the authors of DCE have proposed a post-process that additionally forces *each* element of the pattern to cover at most the chosen proportion of noise. With this post-process the only difference with FENSTER's tolerance to noise relates to the absolute/relative dilemma. Nevertheless, the ability to tolerate a greater number of absent tuples in larger patterns does not affect the quality in this experiment, i. e., FENSTER (see Fig. 11a) and DCE + post-process (see Fig. 11c) extract patterns that cover the exact same tuples.

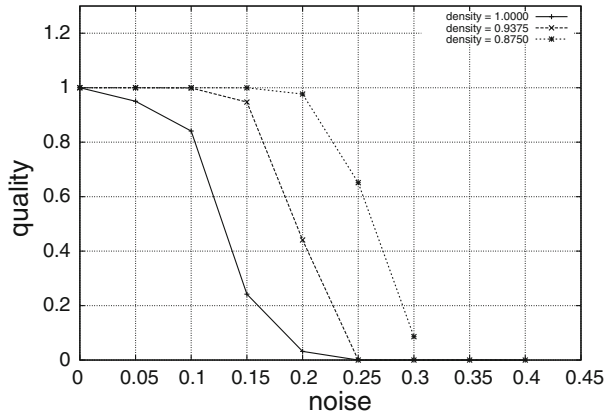
FENSTER and DCE are tested on 4-ary relations. The experimental protocol still follows what Sect. 5.1 details. The hidden patterns contain four elements in every attribute domain (of 16 elements) and the patterns are forced to have at least two elements per attribute. Figure 12 plots the qualities obtained in this setting. Whatever the level of noise, FENSTER (see Fig. 12a) outputs better collections of closed ET-4-sets with $\epsilon = 1$ than without any tolerance to noise (i. e., $\epsilon = 0$). With a noise level of 0.25,



(a) FENSTER.



(b) DCE without post-process.



(c) DCE with post-process.

Fig. 11 Quality of the extracted collections of patterns with at least four elements per attribute in the $32 \times 32 \times 32$ datasets

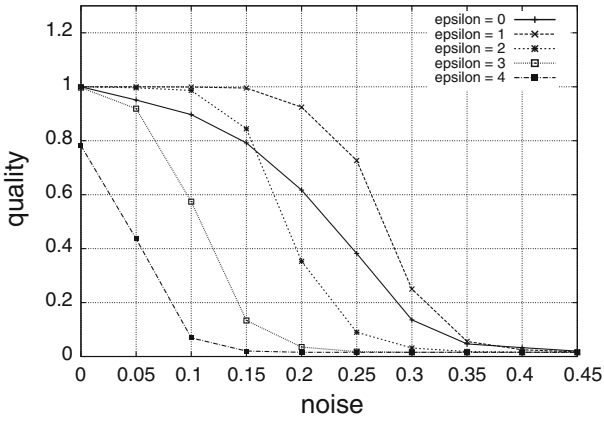
the quality of the collection of exact closed 4-sets is below 0.3, whereas it reaches 0.65 when $\epsilon = 1$. Furthermore, because of the loose minimal size constraints, $\epsilon = 2$ is too high. Comparing FENSTER's results with those of DCE with (Fig. 12c) and without (Fig. 12b) the filtering post-process leads to similar conclusions as those drawn from the 3-dimensional case: the quality of the patterns globally tolerating noise is far beneath that of the patterns tolerating noise per element and, in this latter context, DCE's relative tolerance to noise has almost no effect on the quality w.r.t. FENSTER's absolute tolerance.

The worse quality results of a per-pattern tolerance to noise were explained by the possibility for those patterns to "go beyond" the hidden patterns. The mere quality measure is not enough to justify this explanation. That is why Fig. 13 plots the proportion of the covered tuples that is outside the hidden patterns, i. e., $\frac{|\mathcal{E}\setminus\mathcal{R}_{\text{hidden}}|}{|\mathcal{E}|}$. When the noise is tolerated per-pattern (Fig. 13b) instead of per-element (Fig. 13a), as FENSTER does, this quantity of incorrectly covered tuples clearly is higher. These results relate to the experiment on 4-ary relations but a same conclusion is drawn from the analysis of the patterns extracted in the ternary relations.

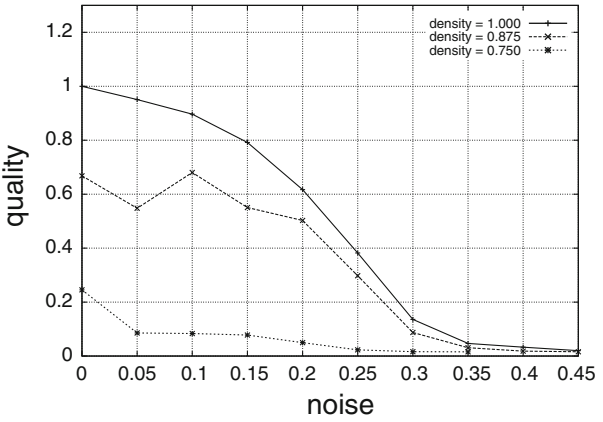
FENSTER and DCE are now seen as binary classifiers of tuples (in the classes "being covered by the hidden patterns" and "not being covered by the hidden patterns") so that ROC curves can be plotted.[6] For a given noise level, such a curve represents the proportion of correctly covered tuples, $\frac{|\mathcal{E}\cap\mathcal{R}_{\text{hidden}}|}{|\mathcal{R}_{\text{hidden}}|}$, in function of the proportion of incorrectly covered tuples, $\frac{|\mathcal{E}\setminus\mathcal{R}_{\text{hidden}}|}{|(\times_{i=1}^{n}\mathcal{D}^i)\setminus\mathcal{R}_{\text{hidden}}|}$. Figure 14 shows the results obtained from the experiment on 4-ary relations. For more clarity, the curves related to noise levels up to 0.15 were omitted: they all coincide with the leftmost and topmost axises of the plots, i. e., the hidden patterns are entirely covered before any false positive tuple is ever covered. Every number written next to a point corresponds to the related noise tolerance threshold. The ROC curves obtained with a per-element tolerance to noise are similar to those obtained with a per-pattern tolerance to noise. This means that the correctly and incorrectly covered tuples are "ranked" in about the same way when moving from no tolerance to noise (where, obviously, the way to tolerate noise has no influence on the output patterns) to tolerating much noise. Nevertheless, with DCE's per-pattern tolerance, increasing the density threshold more quickly leads to the upper-right corner of the ROC space (one can, for instance, compare the positions of DCE's points relating to a minimal density of 0.875 with those obtained by FENSTER and an absolute tolerance of 1). This reflects the greater sensitivity to parametrization that has already been commented from the quality results. This difference is quite important in practice because, contrary to a standard classification problem, the pattern extraction is unsupervised and the analyst usually has no way to precisely know the level of noise affecting the data, hence the optimal noise tolerance thresholds. The analog analysis of the patterns extracted in the ternary relations leads to the same conclusions.

To limit the false-negative rate, stronger minimal size constraints can be used. By requiring at least five (respectively three) elements per attribute of the ternary (respectively 4-ary) relations, none of the false-positive tuples are ever covered, i. e., the ROC curves coincide with the leftmost axis of the ROC space. Unfortunately, many

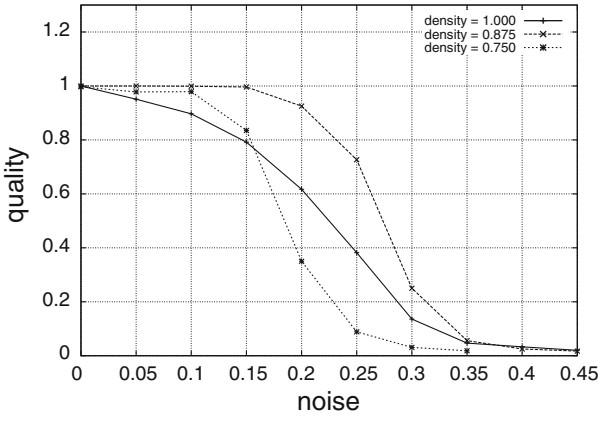
[6] We would like to thank an anonymous reviewer for this original idea.



(a) FENSTER.

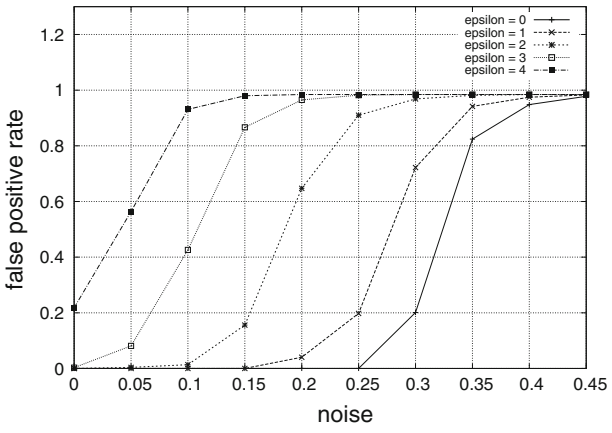


(b) DCE without post-process.

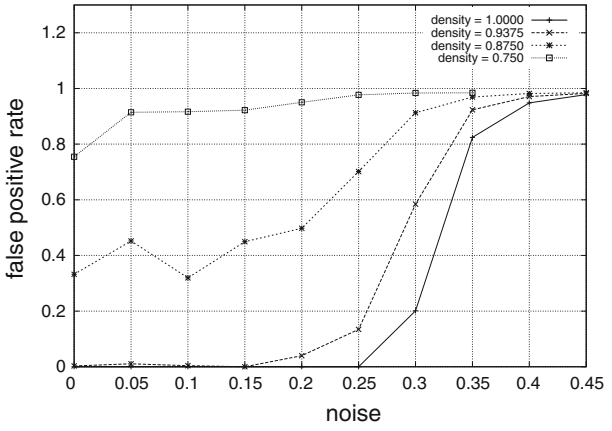


(c) DCE with post-process.

Fig. 12 Quality of the extracted collections of patterns with at least two elements per attribute in the $16 \times 16 \times 16 \times 16$ datasets



(a) FENSTER or DCE with post-process.



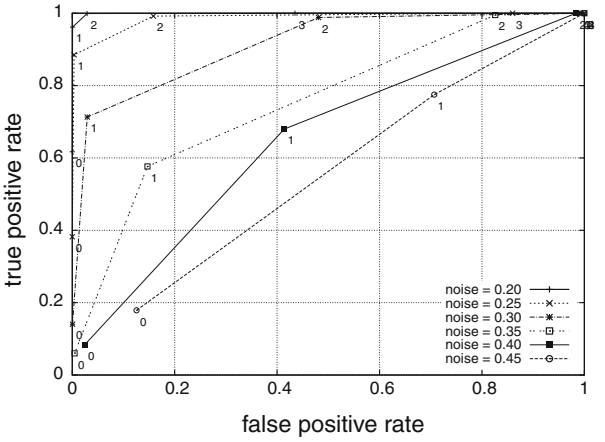
(b) DCE without post-process.

Fig. 13 Proportion of covered 4-tuples outside the hidden patterns for the extractions with at least two elements per attribute in the $16 \times 16 \times 16 \times 16$ datasets

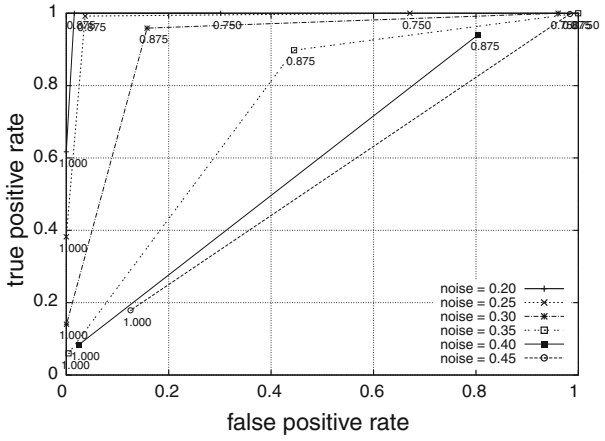
true-positive tuples end up not being covered as well and the quality is worse. In fact, from a noise level of 0.3 (respectively 0.25) and with the greatest noise tolerance we have tested ($\epsilon = 4$), no closed ET-3-set (respectively closed ET-4-set) satisfy the minimal size constraints.

5.3 Size of the output collection of patterns

Figure 15 shows the number of patterns FENSTER and DCE + post-process extract in the synthetic ternary relations. In both cases, these quantities are huge in comparison with the actual number of hidden patterns (4). This confirms what was written to justify the choice of a per-tuple quality measure: numerous fragments of the hidden patterns



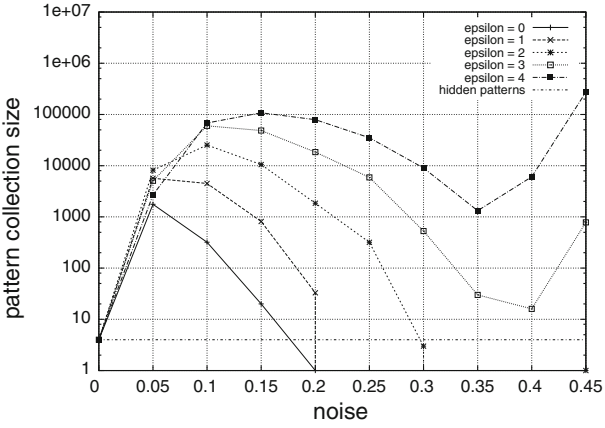
(a) FENSTER or DCE with post-process.



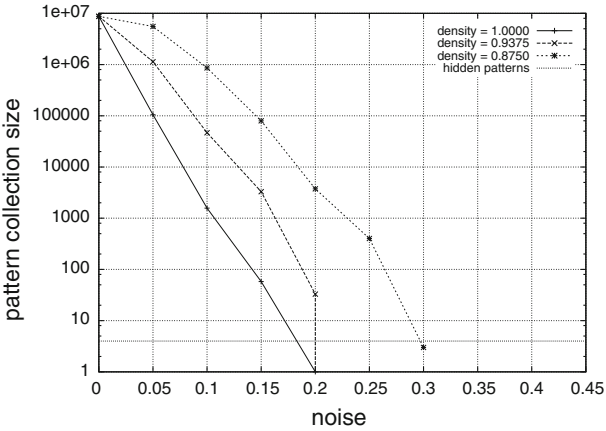
(b) DCE without post-process.

Fig. 14 ROC curves for the extractions with at least two elements per attribute in the $16 \times 16 \times 16 \times 16$ datasets

are discovered. Because these fragments must be agglomerated with a method such as Cerf et al. (2009b) (which has a time complexity that is quadratic in the number of fragments to agglomerate), a smaller number of patterns is preferable so that the agglomeration can be achieved in a reasonable time. In this regard, FENSTER (Fig. 15) appears far preferable to DCE (Fig. 15b) especially in the contexts where little noise affects the data. The reason to this difference is the use (or, for DCE, the lack thereof) of the closedness constraint, which condenses the collection of patterns without any loss of information (see Sect. 2.2). DCE cannot efficiently enforce this constraint because the patterns tolerating noise in a relative way do not satisfy a *downward closure* pro (stated with our terminology: DCE's sub-patterns of a connected closed patterns are not necessarily connected). For instance, when no noise affects the data, DCE not only



(a) FENSTER

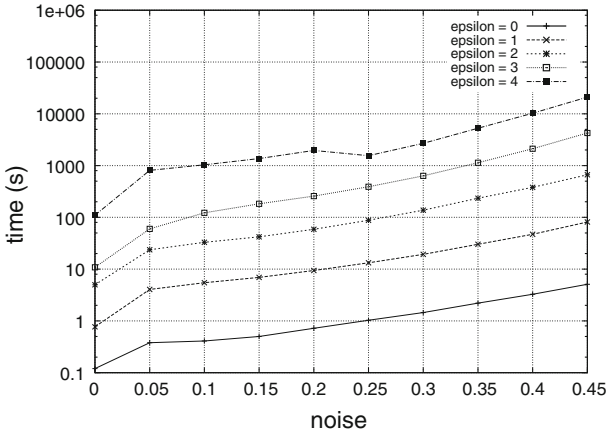


(b) DCE with post-process.

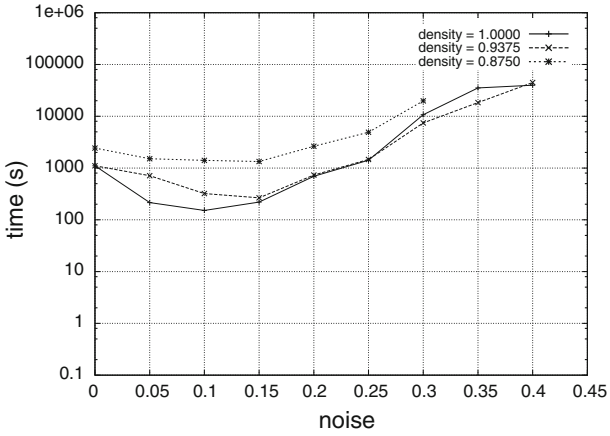
Fig. 15 Number of extracted patterns with at least four elements per attribute in the $32 \times 32 \times 32$ datasets

outputs the four hidden patterns (like FENSTER does) but also all their sub-patterns as long as they have at least four elements per attribute (the minimal size constraints). Naturally, not using DCE's filtering post-process provides even more patterns.

The evolution of the number of closed ET-3-sets when the noise varies is explained by the increasing fragmentation which, at some points makes the patterns so small that they do not satisfy the minimal size constraints. In the very noisy relations, tolerating too much noise ($\epsilon \in \{3, 4\}$) leads to the extraction of patterns covering the false-positive 3-tuples, hence a new increase in the size of the extracted collections. With the 4-ary relations, the results are similar but the latter increase of the number of closed ET-4-sets occurs at lower levels of noise and with less tolerance to noise because the number of 4-tuples outside the hidden (hence, the number of false-positive tuples) is greater.



(a) FENSTER.



(b) DCE with post-process.

Fig. 16 Running times for the extractions with at least four elements per attribute in the $32 \times 32 \times 32$ datasets

5.4 Time performances

The attentive reader has probably noticed that, in the reported results, DCE's tolerance to noise is not "pushed" as much as FENSTER's. In fact, DCE cannot, in a reasonable time (timeout set to 12 h per extraction), complete those extractions. The fastness of FENSTER is its most significant advantage over DCE. Figure 16 plots the running times of these two algorithms running on the synthetic ternary relations. We recall that, at comparable thresholds of noise tolerance, the tuples covered by the patterns are exactly the same when DCE's filtering post-processing is used (see Sect. 5.2). Depending on the setting, FENSTER is between one and four orders of magnitude faster than DCE. DCE's running times without the post-process almost are identical to those with the

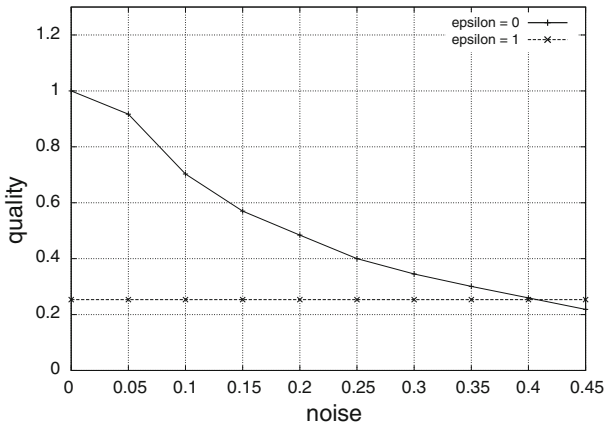
post-process. They actually are slightly worse (probably because of the cost of outputting many more patterns on disk). The reason for FENSTER being much faster than DCE relates to the choice of an absolute tolerance to noise. DCE's relative tolerance to noise does not allow to prune the search space as much because encountering a pattern with a density below the required threshold does mean that none its super-patterns violates this constraint as well. Despite the significant improvement over DCE's efficiency, it must be observed that increasing FENSTER's tolerance to noise leads to an exponential growth of the time it takes to achieve the extraction. This justifies what was previously written about the practical impossibility to tolerate, in a complete way, enough noise to recover the hidden patterns.

The good efficiency of FENSTER is observed in 2-dimensional contexts too. To establish a comparison with `AC-Close`, we could not consider the extraction of patterns with at least four elements per attribute in 32×32 datasets because `AC-Close` (parametrized with $\alpha = 0.75$, i. e., the *exact* support of the patterns must exceed $0.75 \times 4 = 3$ elements) could not terminate in a reasonable time. That is why the subsequent experiments deal with 16×16 datasets containing four overlapping 4×4 patterns. The closed patterns extracted by both FENSTER and `AC-Close` are constrained to contain at least two elements per attribute[7]. α is set to 0.5 and various (relative) noise tolerance levels ϵ are tested. Like with FENSTER, such a level is applied to both attributes.

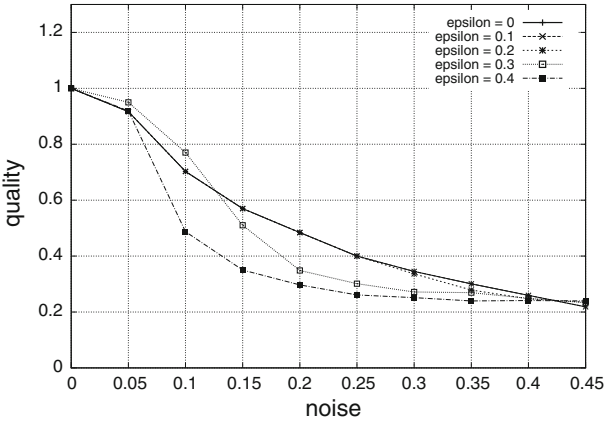
Figure 17 plots the quality results of FENSTER and `AC-Close`. Because the hidden patterns are small, the best results are obtained with no tolerance to noise. In this setting, `AC-Close` and FENSTER compute the same collections of patterns. It is noticeable that a relative tolerance to noise allows more subtle variations of the returned collections. Figure 18 compares the time performances of FENSTER (with $\epsilon = 0$, FENSTER's extractions take less than 0.01 s and are not reported) and `AC-Close`. On this small relation, FENSTER already runs up to three orders of magnitude faster than `AC-Close`. As mentioned earlier, this difference increases with the size of the dataset (even if the same ratio `size of an attribute domain / minimal size constraint` is kept). The reason why FENSTER runs significantly faster than `AC-Close` is again related to the choice for an absolute tolerance to noise. The steep increase of `AC-Close`'s running times when the noise level grows indicates a poor scalability w.r.t. the number of tuples (the hidden patterns only cover a small part of the $16 \times 16 = 256$ tuples, therefore the noise mainly is false-positive and its level roughly is the density of the dataset). Interestingly, `AC-Close`'s performances do not seem to be affected by the chosen tolerance to noise (the curves for $\epsilon \in \{0, 0.1, 0.2, 0.3, 0.4\}$ all coincide).

To further study the scalability of FENSTER, Fig. 19 shows its running times when one of the attribute domain grows (real datasets rarely are cubic). The considered datasets are $32 \times 32 \times 32r$ large and r varies from 10 to 100 by increments of 10. The rightmost point therefore relates to an extraction in a $32 \times 32 \times 3200$ relation. There always are four patterns of size $8 \times 8 \times 8$ affected by 10 of noise. The false-

[7] `AC-Close` cannot enforce a minimal size constraint on both attributes. As a consequence the minimal size constraint on one of them is enforced in a post-processing step. Without this post-process, worse quality measures are obtained.



(a) FENSTER.

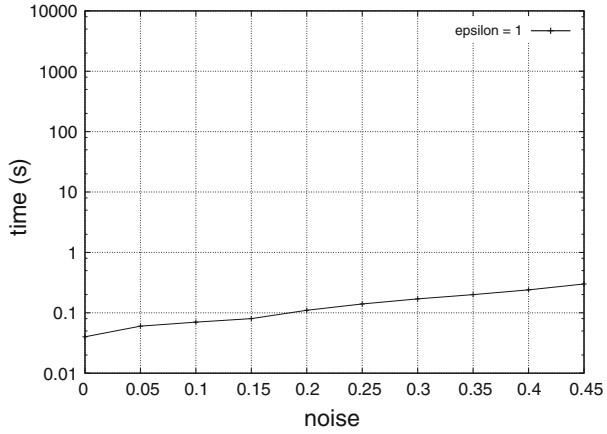


(b) `AC-Close`.

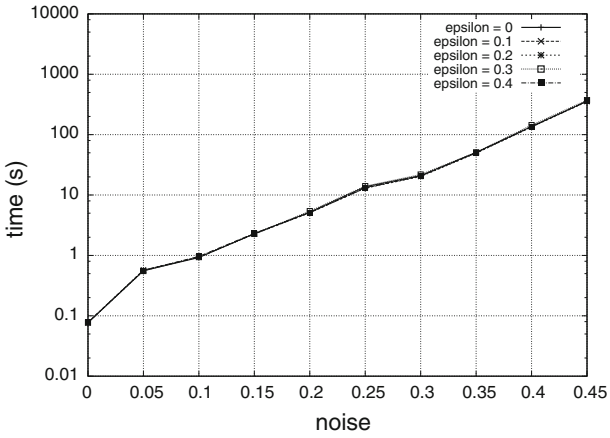
Fig. 17 Quality of the extracted collections of patterns with at least two elements per attribute in the 16×16 datasets

positive noise therefore becomes more and more problematic when r increases. Even like that, all the extracted closed ET-3-sets with $\epsilon = 2$ exactly cover the hidden patterns, i. e., the quality always is 1. FENSTER's running time exponentially increases with r. This is expected since the minimal size constraint on the growing attribute domain is kept constant, hence weaker and weaker (with patterns that grow in proportion with the dataset, one can simply rely on random sampling to decrease the workload).

Fairly testing the scalability w.r.t. the arity of the relation is a hard task because some other characteristics of the relations cannot be kept constant. Using cubic datasets, we decided to fix the number of (absent and present) tuples to 4,096 and to extract, with $\epsilon = 2$, closed ET-n-sets containing at least one fourth of the number of elements per attribute. For instance, the 2-dimensional relations have 64 elements per attribute



(a) FENSTER.



(b) AC-Close.

Fig. 18 Running times for the extractions with at least two elements per attribute in the 16×16 datasets

$(64 \times 64 = 4096)$ and the extracted closed ET-2-sets must contain at least 16 of them $(\frac{16}{64} = \frac{1}{4})$. The relations are only composed of false-positive noise to not introduce more parameters that cannot be kept constant when the arity changes. Figure 20 shows FENSTER's running times when the noise level (in this experiment, it is as well the density of the relations) varies from 0.05 to 0.45 in 2, 3, 4 and 6-ary relations. The worst results are obtained with the 2-dimensional and 6-dimensional datasets. However, it is hard to draw conclusions given the unavoidable imperfections of the setting: the 2-dimensional case appears harder because only two minimal size constraints prune the search space, whereas the 6-dimensional setting leads to the extraction of tens of thousands of patterns that cover every present tuple (one element per attribute is sufficient for a pattern to be considered large enough).

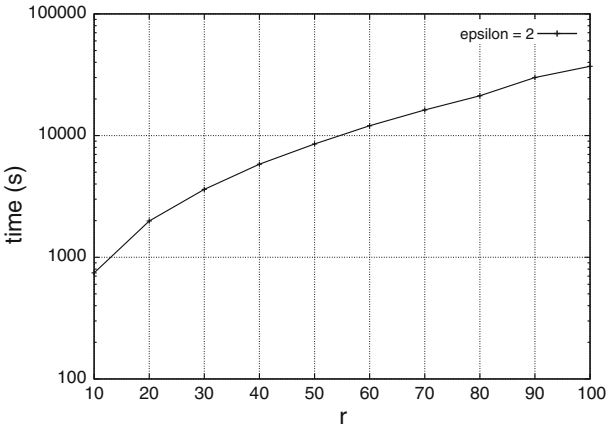


Fig. 19 FENSTER's running times for the extractions, with $\epsilon = 2$, in $32 \times 32 \times 32r$ datasets

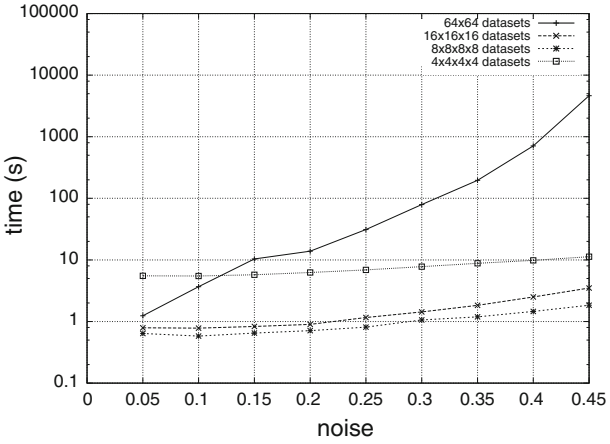


Fig. 20 FENSTER's running times for the extractions, with $\epsilon = 2$, in 2, 3, 4 and 6-dimensional datasets with 4,096 (absent and present) tuples

6 Mining a real graph evolving in two temporal dimensions

Vélo'v is a bicycle rental service run by the urban community of Lyon, France. Vélo'v stations are spread over Lyon and its environment. At any of these stations, the users can take a bicycle and return it to any other station. Whenever a bicycle is rented or returned, this event is logged. We obtained parts of these logs (e. g., no user identification to preserve privacy) recorded between May 27th 2005 (when the system was opened to the public) and December 17th 2007. They represent more than 13.1 millions rides. The earliest records relate to the users discovering Vélo'v. To study the network usage in "normal" conditions, these earliest records, until December 17th 2005, were ignored. Doing so, two full years are kept and aggregations do not favor any part of the year (along which the network usage evolves). Many records stand for

rides from a station to itself. These rides usually last a few seconds. Among others, they can be explained by users who are not satisfied by the quality of the bicycle they have just rent (e. g., a flat tire). Because, from a given station, the most frequent rides are to itself, keeping these records influence a lot any normalization procedure. That is why these records are removed but, after the post-processing steps, the related routes are all claimed frequent, i. e., appended to the relation. A few more abnormal records (with incoherent dates or with stations that are closed to the public) are removed as well. About 10.2 million records remain after these first steps.

We arbitrarily decide to date a ride with the time when the bicycle was returned. To discover patterns that depend on both this time and the day of the week, the data are aggregated w.r.t. these two scales. More precisely, one directed graph is built per period of time (a one-hour period was chosen) and per day of the week. For instance, one of these graphs presents the rides between nine o'clock and ten o'clock on Mondays. The vertices correspond to the Vélo'v stations. The edges are labeled with weights: the total number of rides from the tail vertex (departure station) to the head vertex (arrival station) during the considered period of time and day of the week. The global activity of the Vélo'v network varies a lot between the different days of the week. For instance, there are 51.3 more rides on Fridays than on Sundays. This difference is even greater between the time periods. For instance, there are about 22 times more rides between 6 and 7 pm than between 5 and 6 am. This global behavior is known. To ignore it, the weights are normalized so that their sum is the same whatever the graph. In this way, when the data are binarized, the related Boolean predicate decides whether routes are frequent w.r.t. the period of time and the day of the week. The distribution of the rides w.r.t. the stations is far from being constant too. One reason is structural. Some stations can contain/receive many more bicycles than others. Because no bicycle can be rented from an empty station and no bicycle can be returned to a full station, the largest stations imply more rides. Furthermore they are better known by the users, who want to minimize the risk of finding an empty or a full station. Another reason is the progressive installation of the stations. In December 17th 2005, there were 172 stations in activity. In December 17th 2007, they were 315. Because some stations were closed, there are 327 different stations in the dataset. Obviously a station that opened little before December 17th 2007 cannot be involved in as many records as another one that has been in activity since the beginning. A local binarization partially handles these differences. The computation of a p-value inspires the details of this technique. It considers the vertices one by one, computes the sum S of the weights of both its incoming and outgoing edges, and claims frequent the routes related to the edges with the greatest values and whose sum is just beyond $0.1 \times S$. By definition, this procedure keeps at least one edge involving each station. In average, 191 edges per station are kept (still excluding the reflexive routes). The resulting 4-ary relation, namely $\mathcal{R}_{\text{Vélo'v}}$, contains 117,411 4-tuples (including the reflexive routes, which were previously put to one side), hence a $\frac{117{,}411}{7 \times 24 \times 327 \times 327} = 0.7\%$ density. Table 3 summarizes some of its characteristics and lists the minimal sizes that the extracted patterns are constrained to have in the subsequent experiments.

Furthermore, to focus on the most interesting closed ET-4-sets, the specificities of the dataset are translated to additional constraints. Indeed, $\mathcal{R}_{\text{Vélo'v}}$ is not only a 4-ary relation but a dynamic graph too. In this context, ET-4-sets which are symmetric w.r.t.

Table 3 Characteristics of $\mathcal{R}_{\text{Vélo'v}}$ and minimal sizes of the mined patterns

Number of days of the week	7
Number of 1-h periods of time	24
Number of (departure and arrival) stations	327
Number of present tuples	117, 411
Density	0.7 %
Minimal number of days in a pattern	2
Minimal number of 1-h periods in a pattern	3
Minimal number of departure stations in a pattern	3
Minimal number of arrival stations in a pattern	3

Table 4 Number of patterns, without the almost-contiguity constraint, in $\mathcal{R}_{\text{Vélo'v}}$.

	Number of patterns		Running time		Symmetry
	w/o symm.	with symm.	w/o symm.	with symm.	w/o symm. (%)
$\epsilon = (0, 0, 0, 0)$	13	11	3 min 58 s	3 min 49 s	84.62
$\epsilon = (1, 1, 1, 1)$	111	63	1:04 min 09 s	23 min 56 s	54.05
$\epsilon = (3, 2, 2, 2)$	743	342	13:17 min 18 s	2:14 min 24 s	41.05
$\epsilon = (4, 3, 3, 3)$	–	1163	–	3:12 min 34 s	–

the two vertex attributes are particularly interesting: they are quasi-cliques. ET-4-sets that involve contiguous (or almost-contiguous) periods of times are more relevant too. It turns out that these two constraints are piecewise (anti)-monotone (see Cerf et al. (2009c, 2010)). As a consequence, FENSTER efficiently handles them, i. e., can further prune the search space thanks to them. In our experiments, the almost-contiguity constraint is parametrized with a maximal authorized gap of one hour, i. e., the time periods involved in a pattern can be browsed in a sequence where at most one hour separates a period from the next one (e. g., {3–4pm,4–5pm,6–7pm,8–9pm} is a valid set of time periods, whereas {3–4pm,4–5pm,8–9pm} is not).

Tables 4 and 5 list, with and without the symmetry and/or the almost-contiguity constraint(s), the number of patterns in $\mathcal{R}_{\text{Vélo'v}}$ and the time to extract them. They satisfy as well the minimal size constraints indicated in Table 3. The parameters for noise tolerance, given in the first column, follow the order (ϵ^{day}, ϵ^{time}, ϵ^{dep}, ϵ^{arr}). Although it is not mandatory, it usually makes sense to tolerate more absent tuples (i. e., to choose a greater ϵ^i parameter) in a hyperplane of the pattern that is constrained to contain more tuples. In these experiments, for instance, the minimal size constraints force the hyperplane related to a day of the week to contain at least $3 \times 3 \times 3 = 27$ 4-tuples, whereas any orthogonal hyperplane (i. e., related to any of the three other attributes of $\mathcal{R}_{\text{Vélo'v}}$) of a pattern can have as few as $2 \times 3 \times 3 = 18$ 4-tuples. That explains our choice, for a given experiment, of a ϵ^{day} parameter that roughly is $\frac{27}{18} = 1.5$ greater than the other ϵ^i parameters, which are all set to a same value. When much noise is tolerated, the symmetry constraint greatly reduces the running times. Notice also that this gain does not occur to the detriment of the discovery of interesting patterns. Indeed, the closed 4-sets, in $\mathcal{R}_{\text{Vélo'v}}$, naturally are symmetric. The last column of Tables 4 and 5 give, among the closed ET-4-sets extracted without the symmetry constraint, the

Table 5 Number of patterns, with the almost-contiguity constraint, in $\mathcal{R}_{\text{Vélo'v}}$

	Number of patterns		Running time		Symmetry
	w/o symm.	with symm.	w/o symm.	with symm.	w/o symm. (%)
$\epsilon = (0, 0, 0, 0)$	12	10	1 min 52 s	2 min 08 s	83.33
$\epsilon = (1, 1, 1, 1)$	102	57	38 min 22 s	15 min 24 s	55.88
$\epsilon = (3, 2, 2, 2)$	649	264	8:40 min 16 s	1:08 min 43 s	40.68
$\epsilon = (4, 3, 3, 3)$	–	798	–	1:18 min 56 s	–

proportion of them that actually are symmetric. This high proportion decreases when noise is tolerated. This actually is a good effect of the constraint. Indeed, when more noise is tolerated, some "naturally" symmetric patterns are extended with additional departure *or* arrival stations that noise affects (false-positive 4-tuples). The symmetry constraint filters out these poorly relevant patterns. The almost-contiguity constraint has a same effect on the number of output patterns and the time to extract them.

The symmetry and the almost-contiguity constraints not being implemented in DCE, this algorithm was only run with the minimal size constraints. Unfortunately, it does not scale to large relations such as $\mathcal{R}_{\text{Vélo'v}}$. Without any tolerance to noise, it had not output any pattern after 72 h of (unterminated) computation. In contrast, FENSTER only requires 3 min 58 s to discover the 13 closed 4-sets. This shows that FENSTER opens up applicative perspectives that are out of reach of DCE for performance reasons.

By only looking at the number of closed ET-4-sets, that FENSTER outputs, one could believe that a higher tolerance to noise means larger collections, hence a tougher interpretation. This is not really the case. First of all, and as discussed earlier, these numerous patterns can be agglomerated with a method such as Cerf et al. (2009b). Then, even without relying on such a post-process, the analyst can focus on the largest patterns FENSTER outputs. Indeed, the more absent 4-tuples tolerated in the cover of a pattern, the larger the size of the closed ones. As a consequence, greater size constraints can be use. E. g., with $\epsilon = (4, 3, 3, 3)$, only 17 symmetric closed ET-4-sets hold during five days or more. With the same parametrization, only five patterns contain four stations or more. Figures 21 and 22[8] depict two patterns respectively corresponding to these two settings. Larger red dots indicate the geographic positions of the stations involved in the patterns.

The pattern in Fig. 21 involves three stations. One of them (Station 7034) is at the east side of the Rhône river (where many floating bars can be found) and near the university. The two other stations are the largest ones near two places, which are famous for their shops and pubs: the "Opéra" (Station 1002) and the "Place Bellecour" (Station 2001). This pattern indicates that, in the evening, from Monday to Friday (i. e., every working day), many bicycles flow between these three stations. A probable explanation is that most of these rides are done by students who like having a drink after their

[8] These figures were created from OpenStreetMap project data.

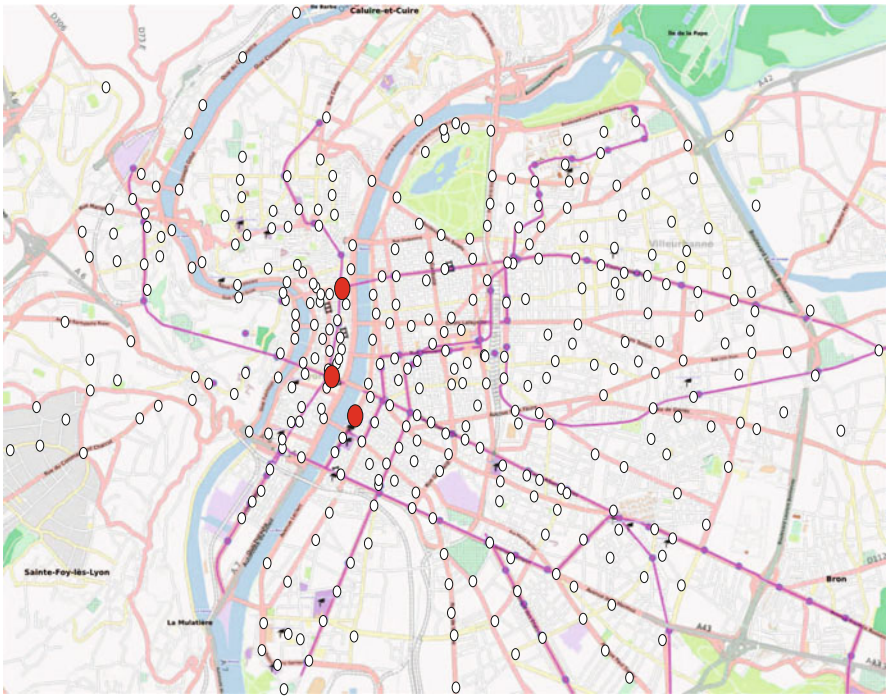


Fig. 21 Every working day, between 5 and 8pm

courses. On the contrary, the pattern in Fig. 22 only holds during the week-ends. Again, the "Opéra" and the "Place Bellecour" are involved. Two other famous places are part of the pattern: the "Saint Jean" cathedral (Station 5004), in the beautiful "Old Lyon", and the main entrance of the largest park in Lyon: the "Parc de la Tête d'Or" (Station 6002). This pattern clearly stands for pleasure trips during the week-end afternoons.

It is interesting to study the effect of a greater tolerance to noise in the discovery of these two patterns. To do so, their Jaccard distances, to every extracted (symmetric) closed ET-4-set, are computed. Table 6 lists the closest patterns for every tested tolerance to noise. When several patterns are at the same Jaccard distance, every variation is given. With no tolerance to noise, the closest closed 4-sets are very different from the ones discovered with $\epsilon = (4, 3, 3, 3)$. The pattern represented in Fig. 21 is missing many days of the week. More precisely, only two days are part of the closest exact closed 4-set. This issue is progressively diminished when more noise is tolerated (three days with $\epsilon = (1, 1, 1, 1)$, four days with $\epsilon = (3, 2, 2, 2)$ and, finally, all five working days with $\epsilon = (4, 3, 3, 3)$). With less tolerance to noise, the pattern represented in Fig. 22 is not only missing elements (in particular the four involved stations are only discovered with $\epsilon = (4, 3, 3, 3)$) but also includes additional elements (in bold letters in Table 6). With no tolerance to noise, the closest discovered pattern actually is not to be related to the one represented in Fig. 22. It is a fragment of another large closed ET-4-set, ({Sat., Sun.}, {3–4pm, 4–5pm, 5–6pm, 6–7pm, 7–8pm}, {Stations 1002, 6002, 3044}), that is extracted with $\epsilon = (4, 3, 3, 3)$.

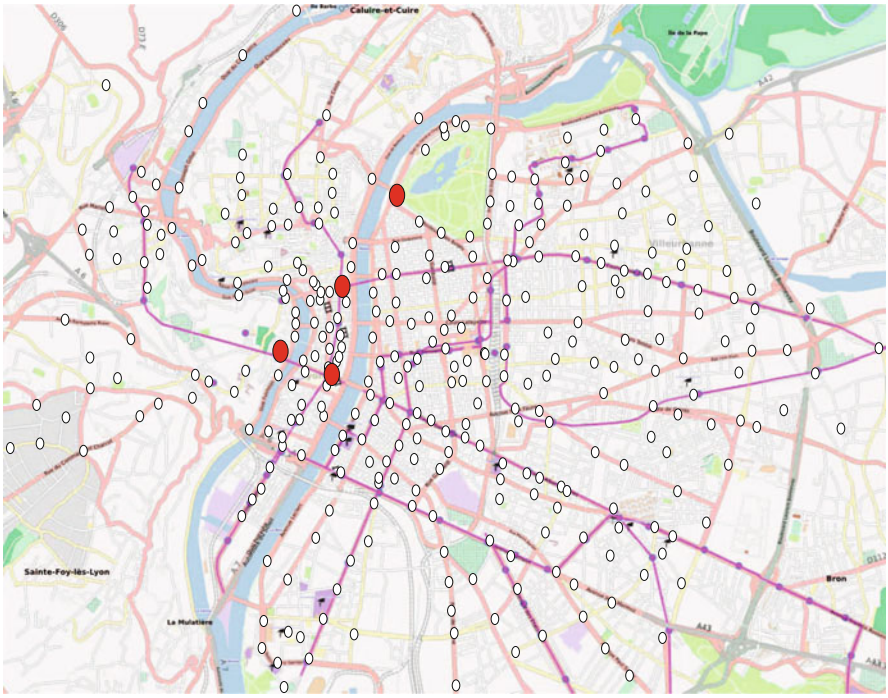


Fig. 22 During the week-ends, between 4 and 7pm

Table 6 Closest patterns to those depicted in Figs. 21 and 22.

$\epsilon = (0, 0, 0, 0)$	({Wed., Thu.}, {5–6pm, 6–7pm, 7–8pm}, {Stations 7034, 1002, 2001}^{2})	({Sat., Sun.}, {5–6pm, 6–7pm, **7–8pm**}, {Stations 1002, 6002, **3044**}^{2})
$\epsilon = (1, 1, 1, 1)$	({Wed., Thu., Mon. (or Tue.)}, {5–6pm, 6–7pm, 7–8pm}, {Stations 7034, 1002, 2001}^{2})	({Sat., Sun.}, {**3-4pm**, 4-5pm, 5-6pm (or 6-7pm)}, {Stations 1002, 2001, 5004}^{2})
$\epsilon = (3, 2, 2, 2)$	({Wed., Thu. and Mon., Tue. or Mon., Fri. or Tue., Fri.}, {5–6pm, 6–7pm, 7–8pm}, {Stations 7034, 1002, 2001}^{2})	({Sat., Sun.}, {**3-4pm**, 4-5pm, 5-6pm, 6-7pm}, {Stations 1002, 5004, 6002}^{2})
$\epsilon = (4, 3, 3, 3)$	({Mon, Tue, Wed., Thu., Fri.}, {5–6pm, 6–7pm, 7–8pm}, {Stations 7034, 1002, 2001}^{2})	({Sat., Sun.}, {4–5pm, 5–6pm, 6–7pm}, {Stations 1002, 2001, 5004, 6002}^{2})

7 Related work

7.1 ET-itemset mining

Several research groups have considered the complete extraction of ET-itemsets in binary relations associating items with transactions (see Gupta et al. (2008) for a survey). None of them is fully satisfactory. A relative tolerance to noise makes the

extraction task very hard and a closedness constraint does not provide a lossless condensation of the ET-itemsets. As a consequence, the extractors tolerating *proportions* of noise, suffer from great scalability issues and they output large collections of ET-itemsets carrying much redundant information. Furthermore, up to, but not including, the recent publication of Poernomo and Gopalkrishnan (2009b), every proposal was either relying on lossy heuristics or imposing additional constraints. Among the extractors relying on lossy heuristics, AFI (Liu et al. 2006) thoroughly demonstrates the semantic necessity to bound the noise inside a pattern with one noise tolerance threshold per domain of the relation (in their case, two thresholds). However, Seppänen and Mannila (2004) pointed earlier the problem of using only one such threshold. Among the approaches that impose additional constraints, `AC-Close` (Cheng et al. 2008) is significant. Indeed, it is the only *closed* ET-itemset extractor with a relative tolerance to noise. Furthermore the experiments show it runs much faster than AFI. These results are possible thanks to a frequency constraint on an *exact* closed itemset every ET-itemset must contain. Thus, `AC-Close` requires an awkward minimal number of transactions *exact* closed itemsets (by opposition to *ET-itemset*) must involve. It extracts them and, in a second step, extends them in closed ET-itemsets. Despite the performance improvement w.r.t. AFI, Sect. 5.4 empirically shows `AC-Close` is intractable on medium-size relations.

Other proposals tolerate, like FENSTER, absolute quantities of absent tuples and, as a consequence, demonstrate a better scalability. `FT-`*Apriori* (Pei et al. 2001) extracts every frequent ET-itemset matching a definition where the noise tolerated per transaction item is bounded (the noise per item is not). To avoid extracting ET-itemsets with some items that almost every transaction (in the pattern) misses, each of these items is forced to hold for at least $\gamma \in \mathbb{N}$ transactions in the pattern. This frequency constraint, restricted to the transactions of the ET-itemset, filters out some of the least relevant patterns. However, because it deals with couples present in (rather than absent from) the relation, ET-itemsets with many transactions may still gather items that many transactions of the pattern do not have. Furthermore, because no closedness constraint is enforced, the extracted collections contain redundant information. VB-FT-Mine (Koh and Yo 2005) builds upon `FT-`*Apriori* and significantly improves both the running times (thanks to the use of bit vectors) and the space requirements (thanks to a depth-first enumeration).

Poernomo and Gopalkrishnan (2007) consider different definitions of an ET-itemset. All of them tolerate noise in an absolute way. To extract them, the BIAS framework enumerates growing subsets of items and words the noise tolerance w.r.t. every enumerated item as an inequality. Integer linear programming allows to derive (one of) the largest set of transactions that satisfies every inequality. Poernomo and Gopalkrishnan (2009) optimize this procedure and generalizes the definition of an ET-itemset. Its transactions are either those that have (or miss) at most, at least, or exactly a given number of items among those in the pattern. The authors propose a recursive equation that computes the number of transactions in an ET-itemset from those of the ET-itemsets with one item less. Beyond the absence of a constraint bounding the noise tolerance w.r.t. items and the absence of a closedness constraint, this approach requires much space. Indeed, it stores all subsets of items that were previously considered and the associated numbers of transactions. On the positive side, the approach can easily be

implemented in any frequent itemset extractor and its running time does not increase much w.r.t. the extraction of *exact* itemsets. Importantly, these two latter contributions output 1-dimensional patterns, i. e., subsets of items for which *there exists* at least one support satisfying the definition. Because several 2-dimensional patterns may involve a same set of items, less patterns are listed and the tuples they cover are not directly accessible.

DR-Miner (Besson et al 2006) extracts complete collections of closed ET-itemsets. Nevertheless, they are differently defined: every transaction (resp. item) "outside" a closed ET-itemset is forced to gather strictly more couples absent from the mined relation than any transaction (resp. item) "inside" it. This prevents the discovery of some ET-itemsets that would be extracted if some additional couples were missing *outside* the pattern. As a consequence, when a hidden pattern is affected by more noise than what DR-Miner can tolerate (while remaining tractable), not only the whole pattern is not discovered (like any other approach) but also most of (if not all) its fragments. Indeed the elements "inside" these fragments are not much different from those "outside" it but inside the hidden pattern.

7.2 ET-n-set mining

When it comes to n-ary relations, DATA- PEELER (Cerf et al. 2009a) extracts every *exact* closed n-set whatever the arity $n \geq 2$. FENSTER reuses some of DATA- PEELER's enumeration principles. However, the fundamental implementation issues pertaining to the enforcement of the newly defined constraints $\mathcal{C}_{\epsilon\text{-connected}}$ and $\mathcal{C}_{\epsilon\text{-closed}}$ and the wiser choice of the element to enumerate at every recursive call make FENSTER differ a lot from DATA- PEELER. The other proposals for exact closed set discovery in the specific and simpler cases of binary and ternary relations are not discussed here (see Cerf et al. (2009a) for such a discussion). Sim et al. (2011) have recently proposed CGQBminer to completely list noise tolerant closed sets in ternary relations seen as collections of bi-partite graphs. These patterns were baptized *maximal cross-graph quasi-bicliques* by analogy with *maximal cross-graph quasi-cliques*. Indeed, the underlying noise tolerance is similar (see Sect. 7.4): in *every* bi-partite graph involved in the pattern, *every* vertex of a quasi-biclique is mostly connected with the others vertices. The dimension gathering graph labels plays a particular role in this definition of noise tolerance. Furthermore, the constraints it imposes on the patterns are quite numerous (one per couple (vertex, bi-partite graph) in the pattern).

DCE (Georgii et al. 2009, 2011) extracts, from real-valued tensors, every *dense n-set*. The constraint "having a density greater than $\alpha \in \mathbb{R}$" relies on the mean of the values covered by a pattern. Such a definition suffers from a relevance problem, which is empirically pointed out in Sect. 5.2: an element can be part of a pattern but disconnected or very weakly connected with the other attributes of the pattern. That is why the authors also define *balanced clusters* in a way that is similar to ET-n-set (balanced clusters tolerate noise in a relative way though) but show that their framework cannot prune the search space with such a definition (DCE's per-element tolerance to noise is enforced in a post-process). Indeed, DCE exploits the (generalized) loose anti-monotonicity (Bonchi and Lucchese 2005) of the density constraint and a reverse

search paradigm (Avis and Fukuda 1996): given a dense n-set, there exists a "sub-pattern" with one element less that is dense. This element, from any attribute domain (hence, the "generalization"), relates to the hyperplane of the pattern with the smallest sum of the real values it contains. Thus, DCE is depth-first and, at every recursive call, enumerates n-sets with such an additional element. In this way, every candidate pattern is traversed once, the minimal density constraint becomes anti-monotone on any enumeration branch and prunes the search space. In this respect, it is alike what Uno (2007) does to list quasi-cliques. These enumerations principles, originating from the choice of a relative tolerance to noise, do not allow the same efficiency results as FENSTER, which ends up running orders of magnitude faster (see Sect. 5.4). Moreover, this same choice of a relative tolerance to noise makes a closedness constraint not provide a lossless condensation of the dense n-sets. Without it, the size of the output collections is problematic (see Sect. 5.3).

7.3 Subspace clustering

Given an n-ary relation, a *subspace cluster* is a *local pattern* (rather than *cluster*) of the form $(X^i)_{i=1..m}$ where every X^i is a subset of a different attribute domain and $m \leq n$. It constrains the data restricted to every *pair* of elements to be *strongly connected*, whereas an ET-n-set imposes one test per element (see Definition 5). Furthermore, in a subspace cluster, every pair of elements (whatever the attribute domain(s) they belong to) must *frequently* appear together in the relation and this frequency is defined w.r.t. the whole relation. More precisely, a subspace cluster is claimed frequent if it exceeds α times its expected value, which assumes a uniform distribution of the n-tuples in the relation. CACTUS (Ganti et al. 1999) and CLICKS (Zaki et al. 2007) extract *maximal* (closedness constraint) subspace clusters from arbitrary n-ary relations. The latter generalizes the former that only mines a restricted class of subspace clusters.

7.4 Cross-graph closed quasi-clique mining

Collections of large graphs were built to help in understanding genetics. These graphs commonly have tens of thousands of nodes and are noisy. For about 5 years, extracting knowledge by crossing such graphs has been a hot topic. For example, there is a need to extract patterns that remain valid across several co-expression graphs obtained from microarray data or to cross the data pertaining to physical interactions between molecules (e. g., protein-protein, protein-gene) with more conceptual data (e. g., co-expression of genes, co-occurrence of proteins in the literature). One of the most promising pattern helping in these tasks is the closed quasi-3-clique. Crochet+ (Jiang and Pei 2009) and Cocain* (Zeng et al. 2007) are the state-of-the-art extractors of closed quasi-3-cliques. They all use the same definition of noise tolerance: every node implied in a pattern must have, in every graph *independently* from the others, a degree exceeding a user-defined proportion of the maximal degree it would reach if the clique was exact. Thus, a pattern involving a subset T of the graphs and a subset N of the nodes needs to satisfy $|T \times N|$ constraints to be a quasi-3-clique, i. e., one constraint *per couple* (timestamp, node). This definition of noise tolerance is different from the

one involved in the definition of the closed ET-n-sets FENSTER extracts. Indeed, in Definition 5, an upper-bounded number of absent n-tuples (rather than a proportion) is tolerated *per element* involved in the pattern, i.e., $(T, N^{\text{tail}}, N^{\text{head}})$ is, by definition, an ET-3-set iff:

- $\forall t \in T$, the dynamic graph contains all 3-tuples in $\{t\} \times N^{\text{tail}} \times N^{\text{head}}$ but $\epsilon^{\text{timestamps}}$ or less.
- $\forall n^{\text{tail}} \in N^{\text{tail}}$, the dynamic graph contains all 3-tuples in $T \times \{n^{\text{tail}}\} \times N^{\text{head}}$ but ϵ^{tail} or less.
- $\forall n^{\text{head}} \in N^{\text{head}}$, the dynamic graph contains all 3-tuples in $T \times N^{\text{tail}} \times \{n^{\text{head}}\}$ but ϵ^{head} or less.

In this way, $(T, N^{\text{tail}}, N^{\text{head}})$ needs to satisfy $|T| + |N^{\text{tail}}| + |N^{\text{head}}|$ constraints to be an ET-3-set. If only symmetric patterns are considered, i.e., $N^{\text{tail}} = N^{\text{head}} = N$, this number becomes $|T| + 2|N|$. In the specific context of undirected graphs (contrary to FENSTER, none of the previously cited approaches can deal with directed graphs), the constraints FENSTER applies on the tails and on the heads are identical. As a consequence, only $|T| + |N|$ constraints defines a symmetric ET-3-set. By comparing this number to $|T \times N|$, it can be written that it is easier for a pattern to be an ET-3-set than a quasi-clique in the sense of Crochet+ or Cocain* (the patterns involving only one timestamp or one node are exceptions to this assertion but they are not very interesting). As a consequence our approach does not scale well to graphs connecting thousands of nodes. Nevertheless, because FENSTER indifferently enumerates timestamps and nodes (no attribute is favored), it can extract closed symmetric ET-3-sets in large collections of smaller graphs, whereas the other algorithms cannot (or they must be used with a very strong minimal size constraint on the number of involved graphs). Finally, (Sim et al. 2011) is a recent proposal that is again less generic than FENSTER (ternary relations only, one particular dimension).

8 Conclusion

Extracting every noise-tolerant itemset in binary relations is a difficult task. Available approaches usually suffer from scalability issues and no closedness constraint is enforced, hence much redundancy in the returned collections. With relations of higher arities, faint noise affects more and more the quality of the patterns. This article presented an algorithm for the complete extraction of noise-tolerant patterns in arbitrary n-ary relations. How much noise is tolerated is parameterized by as many integers as there are attributes in the relation. In every pattern, these integers are upper-bounds of the number of n-tuples involving an element from the related attribute domain, encompassed by the patterns, but absent from the relation. This definition allows a closedness constraint to support a lossless condensation of the extracted collections. Furthermore, and thanks to the incremental computation of counters of absent n-tuples in many subspaces of the relation, the proposed algorithm remains tractable on large relations.

Relevant patterns were discovered in a real-life 4-ary relation involving more than 100,000 tuples and mined under rather loose minimal size constraints. With

the possible enforcement of additional relevance constraints, which even more alleviate the computational requirements, many applicative perspectives are opened up. For instance, we are currently working on the analysis of large amounts of user-generated data on the Web. In such a context, the time is an essential dimension that n-ary relations can take into consideration without having to drop another one. Synthetic datasets have allowed to quantitatively assess the effectiveness and the efficiency of the algorithm. By comparison with its main competitor, which tolerates noise in a different way, this proposal has been shown to extract, orders of magnitude faster, smaller pattern collections of higher quality. In this regard, this contribution increases our understanding of the solution to bring to the unwanted alterations of the relation. Nevertheless a complete approach usually cannot, in a reasonable time, tolerate as much noise as contained in the data. As a consequence, the returned closed patterns remain fragments (though larger fragments than without noise tolerance) of the hidden patterns. To heuristically complement this approach, a hierarchical agglomeration of the patterns (Cerf et al. 2009b) was developed.

Acknowledgements This research has been partially funded by project ANR MDCO Bingo2(2007-2011) and by the FAPEMIG.

References

Avis D, Fukuda K (1996) Reverse search for enumeration. Discret Appl Math 65(1–3):21–46

Bayardo RJ, Goethals B, Zaki MJ (eds) (2004) In: FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004, CEUR Workshop Proceedings, vol 126, CEUR-WS.org

Besson J, Robardet C, Boulicaut JF, Rome S (2005) Constraint-based formal concept mining and its application to microarray data analysis. Intell Data Anal 9(1):59–82

Besson J, Robardet C, Boulicaut JF (2006) Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In: ICCS '06: Proceedings of the 14th International Conference on Conceptual Structures, Springer, pp 144–157

Bonchi F, Lucchese C (2004) On closed constrained frequent pattern mining. In: ICDM '04: Proceedings of the 4th IEEE International Conference on Data Mining, IEEE Computer Society, pp 35–42

Bonchi F, Lucchese C (2005) Pushing tougher constraints in frequent pattern mining. In: PAKDD 05 Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp 114–124

Boulicaut JF, Bykowski A (2000) Frequent closures as a concise representation for binary data mining. In: PAKDD '00: Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp 62–73

Calders T, Rigotti C, Boulicaut JF (2005) A survey on condensed representations for frequent sets. In: Constraint-Based Mining and Inductive Databases, Springer, Lecture Notes in Computer Science, vol 3848, pp 64–80

Cerf L, Besson J, Robardet C, Boulicaut JF (2009a) Closed patterns meet n-ary relations. ACM Trans Knowl Discov Data 3(1):1–36

Cerf L, Mougel PN, Boulicaut JF (2009b) Agglomerating local patterns hierarchically with ALPHA. In: CIKM '09: Proceedings of the 18th International Conference on Information and Knowledge Management, ACM Press, pp 1753–1756

Cerf L, Nguyen TBN, Boulicaut JF (2009c) Discovering relevant cross-graph cliques in dynamic networks. In: ISMIS '09: Proceedings of the 18th International Symposium on Methodologies for Intelligent Systems, Springer, pp 513–522

Cerf L, Nguyen TBN, Boulicaut JF (2010) Mining constrained cross-graph cliques in dynamic networks. Inductive databases and constraint-based data mining, Springer, pp 199–228

Cheng H, Yu PS, Han J (2008) Approximate frequent itemset mining in the presence of random noise. In: Maimon O, Rokach L (eds) Soft computing for knowledge discovery and data mining. Springer, pp 363–389

Gallo A, Bie TD, Cristianini N (2007) MINI: Mining informative non-redundant itemsets. In: PKDD '07: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Springer, pp 438–445

Gallo A, Mammone A, Bie TD, Turchi M, Cristianini N (2009) From frequent itemsets to informative patterns. Tech. Rep. 123936, University of Bristol, UK

Ganter B, Stumme G, Wille R (eds) (2005) Formal concept analysis, foundations and applications, lecture notes in computer science. Springer, Berlin

Ganti V, Gehrke J, Ramakrishnan R (1999) CACTUS: clustering categorical data using summaries. In: KDD '99: Proceedings of the 5th SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, pp 73–83

Garriga GC, Kralj P, Lavrac N (2008) Closed sets for labeled data. J Mach Learn Res 9:559–580

Georgii E, Tsuda K, Schölkopf B (2009) Multi-way set enumeration in real-valued tensors. In: DMMT '09: Proceedings of the 2nd ACM SIGKDD Workshop on Data Mining using Matrices and Tensors, ACM Press, pp 32–41

Georgii E, Tsuda K, Schölkopf B (2011) Multi-way set enumeration in weight tensors. Mach Learn 82(2):123–155

Goethals B (2010) Frequent set mining. In: Maimon O, Rokach L (eds) Data mining and knowledge discovery handbook. Springer, Berlin pp 321–338

Gupta R, Fang G, Field B, Steinbach M, Kumar V (2008) Quantitative evaluation of approximate frequent pattern mining algorithms. In: KDD '08: Proceedings of the 14th SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, pp 301–309

Jaschke R, Hotho A, Schmitz C, Ganter B, Stumme G (2006) TRIAS: an algorithm for mining iceberg tri-lattices. In: ICDM '06: Proceedings of the 6th IEEE International Conference on Data Mining, IEEE Computer Society, pp 907–911

Ji L, Tan KL, Tung AKH (2006) Mining frequent closed cubes in 3D data sets. In: VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB Endowment, pp 811–822

Jiang D, Pei J (2009) Mining frequent cross-graph quasi-cliques. ACM Trans Knowl Discov Data 2(4):1–42

Koh JL, Yo PW (2005) An efficient approach for mining fault-tolerant frequent patterns based on bit vector representations. In: DASFAA '05: Proceedings of the 10th International Conference on Database Systems for Advanced Applications, Springer, pp 568–575

Liu J, Paulsen S, Sun X, Wang W, Nobel AB, Prins J (2006) Mining approximate frequent itemsets in the presence of noise: algorithm and analysis. In: SDM '06: Proceedings of the 6th SIAM International Conference on Data Mining, SIAM, pp 405–416

Pan F, Cong G, Tung AK, Yang J, Zaki MJ (2003) CARPENTER: finding closed patterns in long biological datasets. In: KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, pp 637–642

Pasquier N, Bastide Y, Taouil R, Lakhal L (1999) Efficient mining of association rules using closed itemset lattices. Inf Syst 24(1):25–46

Pei J, Tung AKH, Han J (2001) Fault-tolerant frequent pattern mining: problems and challenges. In: DMKD '01: Proceedings of the 6th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, ACM Press

Poernomo AK, Gopalkrishnan V (2007) Mining statistical information of frequent fault-tolerant patterns in transactional databases. In: ICDM '07: Proceedings of the 7th IEEE International Conference on Data Mining, IEEE Computer Society, pp 272–281

Poernomo AK, Gopalkrishnan V (2009a) Efficient computation of partial-support for mining interesting itemsets. In: SDM '09: Proceedings of the 9th SIAM International Conference on Data Mining, SIAM, pp 1014–1025

Poernomo AK, Gopalkrishnan V (2009b) Towards efficient mining of proportional fault-tolerant frequent itemsets. In: KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, pp 697–706

Seppänen JK, Mannila H (2004) Dense itemsets. In: KDD '04: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, pp 683–688

Sim K, Liu G, Gopalkrishnan V, Li J (2011) A case study of financial ratios via cross-graph quasi-cliques. Inf Sci 181:201–216

Stumme G, Taouil R, Bastide Y, Pasquier N, Lakhal L (2002) Computing iceberg concept lattices with titanic. Data Knowl Eng 42(2):189–222

Uno T (2007) An efficient algorithm for enumerating pseudo cliques. In: ISAAC '07: Proceedings of the 18th International Symposium on Algorithms and Computation, Springer, pp 402–414

Yang C, Fayyad U, Bradley PS (2000) Efficient discovery of error-tolerant frequent itemsets in high dimensions. Tech. Rep. 2000-20, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

Zaki MJ (2004) Mining non-redundant association rules. Data Min Knowl Discov 9(3):223–248

Zaki MJ, Peters M, Assent I, Seidl T (2007) CLICKS: an effective algorithm for mining subspace clusters in categorical datasets. Data Knowl Eng 60(1):51–70

Zeng Z, Wang J, Zhou L, Karypis G (2007) Out-of-core coherent closed quasi-clique mining from large dense graph databases. ACM Trans Database Syst 32(2):13–42