Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference (FLAIRS 2010)

Optimized Mining of a Concise Representation for Frequent Patterns Based on Disjunctions Rather than Conjunctions *

Tarek Hamrouni
URPAH, Faculty of Sciences of Tunis, Tunisia and CRIL-CNRS, Artois University, France.
{tarek.hamrouni@fst.rnu.tn, hamrouni@cril.univ-artois.fr}

Sadok Ben Yahia
URPAH, Faculty of Sciences of Tunis, Tunisia.
sadok.benyahia@fst.rnu.tn

Engelbert Mephu Nguifo
LIMOS-CNRS, Blaise Pascal University, France.
engelbert.mephu_nguifo@univ-bpclermont.fr

Abstract

Exact condensed representations were introduced in order to offer a small-sized set of elements from which the faithful retrieval of all frequent patterns is possible. In this paper, a new exact concise representation only based on particular elements from the disjunctive search space will be introduced. In this space, a pattern is characterized by its disjunctive support, *i.e.*, the frequency of complementary occurrences – instead of the ubiquitous co-occurrence link – of its items. In this respect, we mainly focus here on proposing an efficient tool for mining this representation. For this purpose, we introduce an algorithm, called DSSRM, dedicated to this task. We also propose several techniques to optimize its mining time and memory consumption. The empirical study carried out on benchmark datasets shows that DSSRM is faster by several orders of magnitude than MEP; the mining algorithm of the unique other representation using disjunctions of items.

Introduction and motivations

Given a set of items and a set of transactions, the well known frequent pattern mining problem consists of getting out, from a dataset, patterns having a number of occurrences (*i.e.*, conjunctive support or simply support) greater than or equal to a user-defined threshold (Agrawal and Srikant 1994). Unfortunately, in practice, the number of frequent patterns is overwhelmingly large, hampering its effective exploitation by end-users. In this situation, a determined effort focused on defining a manageably-sized set of patterns from which we can regenerate all frequent patterns along with their exact supports. Such a set is commonly called *exact condensed* (*or concise*) *representation for frequent patterns*. On the other hand, in many real-life applications like market basket analysis, medical data analysis, social network analysis and bioinformatics, etc., the disjunctive connector linking items can bring key information as well as a summarizing method of conveyed knowledge. An interesting solution is then offered through the concise representation based on the joint use of some particular disjunctive patterns – more precisely essential patterns (Casali, Cicchetti, and Lakhal 2005; Kryszkiewicz 2009) and disjunctive closed patterns (Hamrouni, Ben Yahia, and Mephu Nguifo 2009). This representation constitutes a basis for straightforwardly deriving the conjunctive, disjunctive and negative frequencies of a pattern without information loss. It is also characterized by interesting compactness rates when compared to the representations of the literature (please see (Hamrouni, Ben Yahia, and Mephu Nguifo 2009) for details).

From an algorithmic point of view, the MEP algorithm (Casali, Cicchetti, and Lakhal 2005) was proposed for extracting the frequent essential pattern-based representation. On the other hand, only little attention was paid to the mining performances of the disjunctive closed pattern-based representation, since the quantitative aspect was the main thriving focus in (Hamrouni, Ben Yahia, and Mephu Nguifo 2009). In this paper, for the sake of filling this gap, we mainly focus on a new algorithm, called DSSRM,[1] dedicated to an optimized extraction of disjunctive closed patterns. To the best of our knowledge, this algorithm is the first one aiming at mining disjunctive patterns through a dedicated traversal of the disjunctive search space. The DSSRM algorithm hence relies on an efficient method based on an exploitation of the complementary of a pattern w.r.t. the set of items of the dataset. We also propose a thorough discussion about several other optimization techniques used to speed it up. Then, we carry out series of experiments on real-life datasets in order to: ***(i)*** Assess the impact of the optimization of the disjunctive support computation adopted by DSSRM. ***(ii)*** Compare both the DSSRM and MEP algorithms performances. The obtained results assert, on the one hand, the added-value of the optimization techniques we introduced and, on the other hand, that DSSRM outperforms MEP by several orders of magnitude.

The remainder of the paper is organized as follows. Section 2 presents the key notions used in this paper. In Section 3, we describe the main ideas of the proposed representation. Section 4 details the DSSRM algorithm, dedicated to the extraction of this representation. Several optimization techniques designed in order to improve its computational time and memory consumption are also described. In Sec-

*This work was partially supported by the French-Tunisian project CMCU-Utique 05G1412.

[1]DSSRM is the acronym of Disjunctive Search Space-based Representation Miner.

tion 5, an experimental study shows the utility of the proposed approach. Section 6 concludes the paper and points out our future work.

Key notions

In this section, we briefly present the key notions that will be of use throughout the paper.

Definition 1 (***Dataset***) *A dataset is a triplet $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ where $\mathcal{T}$ and $\mathcal{I}$ are, respectively, a finite set of transactions and items, and $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{I}$ is a binary relation between the transactions and items. A couple $(t,i) \in \mathcal{R}$ denotes that the transaction $t \in \mathcal{T}$ contains the item $i \in \mathcal{I}$.*

Example. *We will consider in the remainder an example dataset $\mathcal{D}$ composed by the set of transactions $\mathcal{T}$ = $\{$(1, ABCD), (2, ACDE), (3, ACEF), (4, ABDEF), (5, BCDF)$\}$.*(2)

The following definition presents the different types of supports that can be assigned to a pattern.

Definition 2 (***Supports of a pattern***) *Let $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ be a dataset and I be a pattern. We mainly distinguish three kinds of supports related to I:*

$$\begin{aligned} Supp(I) &= \mid \{t \in \mathcal{T} \mid (\forall\, i \in I, (t,\, i) \in \mathcal{R})\} \mid \\ Supp(\vee I) &= \mid \{t \in \mathcal{T} \mid (\exists\, i \in I, (t,\, i) \in \mathcal{R})\} \mid \\ Supp(\overline{I}) &= \mid \{t \in \mathcal{T} \mid (\forall\, i \in I, (t,\, i) \notin \mathcal{R})\} \mid \end{aligned}$$

Example. *Supp*(BC) = $\mid \{1, 5\} \mid$ = **2**, *Supp*($\vee$ BC) = $|\{1, 2, 3, 4, 5\}|$ = **5** *and Supp*($\overline{\text{BC}}$) = **0**.

In the remainder, if there is no risk of confusion, the conjunctive support will simply be called support. Having the disjunctive supports of patterns subsets, we can derive their conjunctive supports using an inclusion-exclusion identity (Galambos and Simonelli 2000), while their negative supports are derived thanks to the De Morgan law, as follows:

Lemma 3 (***Relations between the supports of a pattern***) *Let I be a pattern. The following equalities hold:*

$$Supp(I) = \sum_{\emptyset \subset I_1 \subseteq I} (-1)^{|I_1|-1} Supp(\vee I_1) \quad (1)$$

$$Supp(\overline{I}) = \mid \mathcal{T} \mid - Supp(\vee I) \quad (2)$$

Example. *Given the respective disjunctive supports of* BC' *subsets, its conjunctive and negative supports are inferred as follows:*

- *Supp*(BC) = $(-1)^{|\text{BC}|-1}$ *Supp*($\vee$BC) + $(-1)^{|\text{B}|-1}$ *Supp*($\vee$B) + $(-1)^{|\text{C}|-1}$ *Supp*($\vee$C) = - *Supp*($\vee$BC) + *Supp*($\vee$B) + *Supp*($\vee$C) = - **5** + **3** + **4** = **2**.
- *Supp*($\overline{\text{BC}}$) = $|\mathcal{T}|$ - *Supp*($\vee$BC) = **5** - **5** = **0**.

A frequent (essential) pattern is defined as follows.

Definition 4 (***Frequent, Essential, Frequent essential pattern***) *For a user-defined minimum support threshold minsupp, a pattern I is frequent if $Supp(I) \geq$ minsupp. I is said to be essential if $Supp(\vee I) > \max\{Supp(\vee I \backslash \{i\}) \mid i \in I\}$ (Casali, Cicchetti, and Lakhal 2005). I is a frequent essential pattern if it is simultaneously frequent and essential.*

Example. *Consider our running dataset. For minsupp* = **2**, AC *is a frequent essential pattern. Indeed,* AC *is a frequent pattern since Supp*(AC) = **3** $\geq$ *minsupp. Moreover,* AC *is an essential pattern since Supp*($\vee$AC) = **5** *is strictly greater than* $\max\{Supp(\vee \text{A}),\ Supp(\vee \text{C})\}$*, equal to* **4**.

We denote by $\mathcal{FP}$ (*resp.* $\mathcal{FEP}$) the set of frequent patterns (*resp.* frequent essential patterns). In (Casali, Cicchetti, and Lakhal 2005), it was proved that $\mathcal{FEP}$ is downward closed. Indeed, if I is a frequent essential pattern then all its subsets are also frequent essential ones. To ensure the exact regeneration process of frequent patterns, the authors of (Casali, Cicchetti, and Lakhal 2005) proposed to augment this set by maximal frequent patterns. This obliges the associated mining algorithm, namely MEP (Casali, Cicchetti, and Lakhal 2005), to bear the cost of exploring ***both*** disjunctive and conjunctive search spaces to mine both sets composing the representation. Indeed, essential patterns and maximal frequent patterns are respectively characterized by different kinds of supports: disjunctive for the former and conjunctive support for the latter. The MEP algorithm thus relies on a dedicated algorithm for exploring the conjunctive search space through the invocation of an external algorithm for mining maximal frequent patterns.

Disjunctive closed patterns

Structural properties

Given an arbitrary pattern, its disjunctive closure is equal to the maximal pattern, w.r.t. set inclusion, containing it and having the same disjunctive support. The following definition formally introduces the disjunctive closure.

Definition 5 (***Disjunctive closure of a pattern***) *The disjunctive closure of a pattern I is: $h(I) = \max_{\subseteq}\{I_1 \subseteq \mathcal{I} \mid (I \subseteq I_1) \wedge (Supp(\vee I) = Supp(\vee I_1))\} = I \cup \{i \in \mathcal{I} \backslash I \mid Supp(\vee I) = Supp(\vee (I \cup \{i\}))\}$.*

Example. *Consider our running dataset. We have h(*D*) =* BD*. Indeed,* BD *is the maximal pattern containing* D *and having a disjunctive support equal to that of* D.

The closure operator h induces an equivalence relation on the power-set of $\mathcal{I}$, which partitions it into disjoint subsets called *disjunctive equivalence classes*. The elements of each class have the same disjunctive closure and the same disjunctive support. The essential patterns of a disjunctive equivalence class are the smallest incomparable members, w.r.t. set inclusion, while the disjunctive closed pattern is the largest one.

Disjunctive closed pattern-based representation

Our representation relies on the set defined as follows:

Definition 6 *The set $\mathcal{EDCP}$*(3) *gathers the disjunctive closures of frequent essential patterns: $\mathcal{EDCP} = \{h(I) \mid I \in \mathcal{FEP}\}$.*

An interesting solution ensuring the exactness of the representation based on the disjunctive closed patterns of $\mathcal{EDCP}$ consists in adding the set $\mathcal{FEP}$ of frequent essential patterns. This is stated by the following theorem.

(2) We use a separator-free form for the sets, *e.g.*, ABCD stands for the set of items $\{$A, B, C, D$\}$.

(3) $\mathcal{EDCP}$ is the acronym of Essential Disjunctive Closed Patterns.

Theorem 7 *The set $\mathcal{EDCP} \cup \mathcal{FEP}$ of disjunctive patterns, associated to their respective disjunctive supports, is an exact representation of the set of frequent patterns $\mathcal{FP}$.*

Proof. *Let $I \subseteq \mathcal{I}$. If there is a pattern I_1 s.t. $I_1 \in \mathcal{FEP}$ and $I_1 \subseteq I \subseteq h(I_1)$, then $h(I) = h(I_1)$ since h is isotone as being a closure operator. Hence, Supp($\vee$ I) = Supp($\vee$ I_1). Since the disjunctive support of I is correctly derived, then its conjunctive support can be exactly computed thanks to Lemma 3, and then compared to minsupp in order to retrieve its frequency status. If there is not such a pattern I_1, then I is necessarily encompassed between an infrequent essential pattern and its closure. Consequently, I is infrequent since the frequency is an anti-monotone constraint (Agrawal and Srikant 1994).*

The proof of Theorem 7 can also easily be transformed to a naive algorithm for deriving frequent patterns and their associated supports starting from this representation. This can straightforwardly be done in a levelwise manner that regenerates **1**-frequent patterns, **2**-frequent patterns, and so forth. The $\mathcal{DSSR}$ representation ensures the easy derivation of the disjunctive support of each frequent pattern, and hence its negative one using the De Morgan law as well as its conjunctive support through only to evaluation of a unique inclusion-exclusion identity. Another important feature of this representation is the fact that it is homogeneous in the sense that it is ***only*** composed by disjunctive patterns. It hence requires ***only*** traversing the disjunctive search space while offering the direct retrieval of the different types of supports of frequent patterns. This representation hence avoids the exploration of the conjunctive search space since it does not need to add supplementary information from the conjunctive search space to check whether a pattern is frequent or not. Since this representation is composed by particular elements within the disjunctive search space, namely essential and disjunctive closed patterns, it will be denoted $\mathcal{DSSR}$ which stands for Disjunctive Search Space-based Representation. Another interesting feature of the $\mathcal{DSSR}$ representation is that it can be stored in a very compact way and without information loss. This is carried out as follows: $\mathcal{DSSR} = \{(e, f\backslash e, Supp(\vee\, e)) \mid e \in \mathcal{FEP}$ and $f = h(e) \in \mathcal{EDCP}\}$. Note that $f\backslash e$ is the difference between f and e. Each disjunctive closed pattern, like f, is then simply derivable by getting the union between e and $f\backslash e$.

The DSSRM algorithm

Hereafter, we introduce the level-wise algorithm DSSRM dedicated to the extraction of the $\mathcal{DSSR}$ representation.

Description

The disjunctive closed patterns composing the $\mathcal{EDCP}$ set have, for associated seeds, the set $\mathcal{FEP}$ of frequent essential patterns. Interestingly enough, these latter seeds form a downward closed set. Thus, a levelwise traversal of the search space is indicated for localizing them without overhead. The DSSRM algorithm is thus designed to adopt such a traversal technique for localizing the required seeds. Once located, their disjunctive closure will be efficiently derived as explained hereafter. In this respect, the computation of the disjunctive closures of equal-size patterns can be performed using a unique pass over the dataset.

According to Definition 5, a naive method for obtaining the closure of I is to augment it by the items maintaining its support unchanged. However, this requires knowing beforehand the disjunctive support of $(I \cup \{i\})$ for each item $i \in \mathcal{I}\backslash I$, what can be very costly. Hence, we propose in the following an efficient method for computing closures. The computation of the disjunctive closures of equal-size patterns can thus be done in a one pass over the dataset using the following lemma.

Lemma 8 (***Structural characterization of the disjunctive closure***) *The disjunctive closure $h(I)$ of a pattern I is the maximal set of items that only appear in the transactions having at least an item of I.*

Thus, the disjunctive closure of I can be computed from the dataset in two steps. First of all, we compute the set $\overline{h(I)}$ of items that appear in the transactions not containing any item of I. Then, by evaluating the set $\mathcal{I}\backslash\overline{h(I)}$, we simply obtain $h(I)$.

Example. *Consider our running dataset. Let us compute the disjunctive closure of the pattern* D. *For this purpose, we need to determine the items that appear in the transactions where* D *is not present. These items are* A, C, E *and* F *w.r.t. the transaction 3, the unique one in which* D *is absent. Thus, we have* $\overline{h(\mathrm{D})}$ = ACEF. *Hence, we obtain* h(D) = $\mathcal{I}\backslash\overline{h(\mathrm{D})}$ = ABCDEF\ACEF = BD. *Note that the computed disjunctive closed pattern of* D *is obviously the same as obtained in Example (cf. page 2).*

By definition, essential patterns are the smallest elements in the disjunctive equivalence classes (*cf.* Definition 4). Therefore, they are the first elements from which the disjunctive closures are computed when a level-wise traversal of the search space is adopted. Dually, the disjunctive closures can be used to efficiently detect essential patterns. This is done by DSSRM thanks to the following proposition.

Proposition 9 (***Characterization of essential patterns based on disjunctive closures***) *Let $I \subseteq \mathcal{I}$ be a pattern. We then have: [I is an essential pattern] $\Leftrightarrow$ [$\forall$ ($I_1 \subset I \wedge |I_1|$ = $|I|$ - 1), $I \nsubseteq h(I_1)$].*

Roughly speaking, I is an essential pattern if it is not included in any disjunctive closure of one of its immediate subsets. This new characterization of essential patterns, adopted by DSSRM, allows the detection of essential patterns without computing their disjunctive supports. Indeed, we only need the disjunctive closures of the immediate subsets of a pattern to guess whether it is essential or not.

The pseudo-code of DSSRM is depicted by Algorithm 1. The associated notations are summarized in Table 1. The mining of the disjunctive closures of $\mathcal{EDCP}$, associated to their disjunctive supports, is carried out using the COMPUTE_SUPPORTS_CLOSURES procedure (*cf.* line 4). To this end, thanks to one pass over the dataset, this procedure computes the conjunctive and disjunctive supports of candidates as well as the complementary, w.r.t. the set of items $\mathcal{I}$, of their associated disjunctive closed patterns. Then, it deduces the disjunctive closures of frequent candidates from their

Notation	Description
$\mathcal{C}_i$ (*resp.* $\mathcal{L}_i$)	: Set of i-candidate (*resp.* i-frequent) essential patterns.
X_i	: Pattern of size i.
$X_i.h$	: Disjunctive closure of X_i.
$X_i.\overline{h}$	: Complementary of $X_i.h$ w.r.t. $\mathcal{I}$: $X_i.\overline{h} = \mathcal{I} \setminus X_i.h$.
$X_i.Conj\_Supp$ (*resp.* $X_i.Disj\_Supp$)	: Conjunctive (*resp.* disjunctive) support of X_i.

Table 1: Notations used in the DSSRM algorithm.

complementary and inserts them in $\mathcal{EDCP}$. On its side, the set $\mathcal{FEP}$ is equal to the union of the different sets of frequent essential patterns of equal size (*cf.* line 5). The generation of $(i + 1)$-candidates is performed by the APRIORI-GEN procedure (Agrawal and Srikant 1994), applied on the retained i-frequent essential patterns (*cf.* line 6). The next instruction (*cf.* line 7) ensures that each element of $\mathcal{C}_{i+1}$ has all its immediate subsets as frequent essential patterns. For this purpose, a candidate having an immediate subset which is not a frequent essential pattern is withdrawn. While pruning non-essential patterns from $\mathcal{C}_{i+1}$ is performed thanks to the characterization of essential patterns using disjunctive closures. Indeed, Proposition 9 allows pruning each candidate included in the disjunctive closure of one of its immediate subsets, since it is necessarily not an essential pattern.

Example. *Consider our running dataset and let minsupp* = **3**. *Initially,* DSSRM *considers the set* $\mathcal{C}_1$ *of* **1**-*essential candidates. For these patterns, the conjunctive and disjunctive supports as well as the complementary of their associated disjunctive closures are computed by the* COMPUTE_SUPPORTS_CLOSURES *procedure thanks to an access to the dataset. The result of this access for the candidates* A *and* B *is shown in Table 2 (Left). Then, this procedure constructs the set* $\mathcal{L}_1$ *containing frequent* **1**-*essential patterns. It also deduces their closures starting from their respective complementary. These closures will be included in* $\mathcal{EDCP}$*, since all items are frequent. Table 2 (Right) shows the result of this step for the candidates* A *and* B.

Thus, $\mathcal{L}_1 = \mathcal{C}_1 = \{A, B, C, D, E, F\}$, $\mathcal{EDCP} = \{$(B, *3*), (C, *4*), (E, *3*), (F, *3*), (AE, *4*), (BD, *4*)$\}$. *Then,* DSSRM *generates the set* $\mathcal{C}_2$ *of the next iteration thanks to the* APRIORI-GEN *procedure. After this step,* $\mathcal{C}_2 = \{$AB, AC, AD, AE, AF, BC, BD, BE, BF, CD, CE, CF, DE, DF, EF$\}$. *In order to only retain essential patterns in* $\mathcal{C}_2$*, the instruction of line 7 is executed to prune the candidates included in a disjunctive closure of one of their proper subsets. The candidates* AE *and* BD *will hence be pruned from* $\mathcal{C}_2$. *This latter set is then reduced to* $\{$AB, AC, AD, AF, BC, BE, BF, CD, CE, CF, DE, DF, EF$\}$, *and the* COMPUTE_SUPPORTS_CLOSURES *procedure will then proceed its elements. The result of the access step for the candidates* AB *and* AC *is shown in Table 3 (Left). After this step, the construction of the sets* $\mathcal{L}_2$ *and* $\mathcal{EDCP}$ *is performed. Infrequent candidates will be located and their closure will not be taken into account since we are only interested in the disjunctive closed patterns associated to frequent essential patterns. This step is shown in Table 3 (Right) for the candidates* AB *and* AC. *The symbol "– –" indicated that* $X_2.h$ *is not computed. Thus,* $\mathcal{L}_2 = \{$AC, AD, AF, CD$\}$. *The set* $\mathcal{EDCP}$ *is augmented by* (ABCDEF, *5*), *while the set* $\mathcal{FEP}$ *is augmented by* $\mathcal{L}_2$. *After that, the* APRIORI-GEN

Algorithm 1: DSSRM

```
Data: A dataset $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$, and the minimum support
      threshold minsupp.
Results: $\mathcal{DSSR} = \mathcal{EDCP} \cup \mathcal{FEP}$.
1  Begin
2      $\mathcal{EDCP} := \emptyset$; $\mathcal{FEP} := \emptyset$; $i := 1$; $\mathcal{C}_1 := \mathcal{I}$;
3      While ($\mathcal{C}_i \neq \emptyset$) do
4          COMPUTE_SUPPORTS_CLOSURES($\mathcal{D}$, minsupp, $\mathcal{C}_i$,
           $\mathcal{L}_i$, $\mathcal{EDCP}$);
5          $\mathcal{FEP} = \mathcal{FEP} \cup \mathcal{L}_i$;
6          $\mathcal{C}_{i+1}$ := APRIORI-GEN($\mathcal{L}_i$);
7          $\mathcal{C}_{i+1} := \{X_{i+1} \in \mathcal{C}_{i+1} \mid \forall\ Y_i \subset X_{i+1}, Y_i \in \mathcal{L}_i$ and
           $X_{i+1} \not\subseteq Y_i.h\}$;
8          $i := i + 1$;
9      return $\mathcal{EDCP} \cup \mathcal{FEP}$;
10 End
11 Procedure COMPUTE_SUPPORTS_CLOSURES($\mathcal{D}$, minsupp,
   $\mathcal{C}_i$, $\mathcal{L}_i$, $\mathcal{EDCP}$)
12 Begin
13     $\mathcal{L}_i := \emptyset$;
14     Foreach ($t \in \mathcal{T}$) do
15         Foreach ($X_i \in \mathcal{C}_i$) do
16             $\Omega := X_i \cap I$ /*$I$ denotes the items associated to the
               transaction $t$*/;
17             If ($\Omega = \emptyset$) then
18                 $X_i.\overline{h} := X_i.\overline{h} \cup I$;
19             Else
20                 $X_i.Disj\_Supp := X_i.Disj\_Supp + 1$;
21                 If ($\Omega = X_i$) then
22                     $X_i.Conj\_Supp := X_i.Conj\_Supp + 1$;
23     Foreach ($X_i \in \mathcal{C}_i$) do
24         If ($X_i.Conj\_Supp \geq$ minsupp) then
25             $\mathcal{L}_i := \mathcal{L}_i \cup \{X_i\}$;
26             $X_i.h := \mathcal{I} \setminus X_i.\overline{h}$;
27             $\mathcal{EDCP} := \mathcal{EDCP} \cup \{(X_i.h, X_i.Disj\_Supp)\}$;
28 End
```

procedure is invoked in order to generate the set $\mathcal{C}_3$ *equal to* $\{$ACD$\}$. *Since* ACD *is included in the closure of its immediate subset* AC, *namely* ABCDEF, *then it is a non-essential pattern. It will hence be pruned. Thus,* $\mathcal{C}_3$ *is empty. Consequently, the iteration process ends. Finally, the* DSSRM *algorithm outputs the exact representation* $\mathcal{DSSR}$.

Theorem 10 ensures the correctness of our algorithm. The associated proof is omitted for lack of available space.

Theorem 10 *The* DSSRM *algorithm is sound and correct. It exactly extracts all the patterns belonging to* $\mathcal{DSSR}$*, associated to their disjunctive supports.*

	Access step			Construction step		
X_1	$Conj\_Supp$	$Disj\_Supp$	$\overline{h}$	$X_1 \in \mathcal{FP}$?	h	$X_1.h \in \mathcal{EDCP}$**?**
A	4	4	BCDF	yes	AE	yes
B	3	3	ACDEF	yes	B	yes

Table 2: The access (Left) and the construction (Right) steps for the **1**-candidates A and B.

	Access step			Construction step		
X_2	$Conj\_Supp$	$Disj\_Supp$	$\overline{h}$	$X_2 \in \mathcal{FP}$?	h	$X_2.h \in \mathcal{EDCP}$**?**
AB	2	5	$\emptyset$	no	$--$	no
AC	3	5	$\emptyset$	yes	ABCDEF	yes

Table 3: The access (Left) and the construction (Right) steps for the **2**-candidates AB and AC.

Optimization techniques

For the sake of improving the efficiency of our algorithm, several optimizations were introduced in its implementation. Indeed, DSSRM relies on the use of efficient data structures (Knuth 1968; Smith 2004) for improving both its performances and memory consumption. For example, a prefix-tree (or trie) is used for storing candidates, while *bitsets* were of use for accelerating AND, OR and NOT operations, and an RB-tree is devoted for an optimized storing of disjunctive closed patterns. In addition, we designed a shrewd method for optimizing the disjunctive support computation that relies on the following theorem:

Theorem 11 *Let $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ be a dataset and $I = \{i_1, \ldots, i_n\} \subseteq \mathcal{I}$ be a pattern. We then have: Supp$(\vee I)$ = $\sum_{j=1}^{n}$ Supp$(\overline{i_1}, \ldots, \overline{i_{j-1}}, i_j)$, where Supp$(\overline{i_1}, \ldots, \overline{i_{j-1}}, i_j)$ = $\mid \{t \in \mathcal{T} \mid ((t,\ i_j) \in \mathcal{R}) \wedge (\forall k \in \{1, \ldots, j-1\}, (t, i_k) \notin \mathcal{R})\} \mid$.*

Proof. (*Sketch*) *The family $\{\{t \in \mathcal{T} \mid ((t,\ i_j) \in \mathcal{R}) \wedge (\forall k \in \{1, \ldots, j-1\}, (t, i_k) \notin \mathcal{R})\}_{j \in \{1,\ldots,n\}}\}$ is a disjoint partition of the set $\{t \in \mathcal{T} \mid \exists\ i \in I,\ (t,\ i) \in \mathcal{R}\}$, whose the cardinality corresponds to the disjunctive support of I.*

Thanks to the aforementioned formula, we devise a new method, called O-METHOD,[4] allowing to efficiently compute *Supp*$(\vee I)$. Our method hence dramatically reduces the number of visited nodes required to compute *Supp*$(\vee I)$. Indeed, contrary to the brute-force method, further denoted BF-METHOD,[5] O-METHOD avoids visiting – in the trie storing candidates – all the children of an item appearing in the considered transaction. In contrary, when using BF-METHOD, for **each** transaction, it requires visiting **all** the items of a pattern candidate. Furthermore, since the family $\{\{t \in \mathcal{T} \mid ((t,\ i_j) \in \mathcal{R}) \wedge (\forall k \in \{1, \ldots, j-1\}, (t, i_k) \notin \mathcal{R})\}_{j \in \{1,\ldots,n\}}\}$ is a disjoint partition of the set of transactions $\{t \in \mathcal{T} \mid \exists\ i \in I,\ (t,\ i) \in \mathcal{R}\}$, a unique support of $\{Supp(\overline{i_1}, \ldots, \overline{i_{j-1}}, i_j)\}_{j \in \{1,\ldots,n\}}$ is updated per transaction. Hence, O-METHOD requires the evaluation of *one* support per transaction.

Experimental results

In this section, we compare, through extensive experiments, the performances of the algorithms allowing the extraction of the representations defined through the disjunctive support, namely those respectively based on disjunctive closed patterns and frequent essential ones.[6] For the former, two versions of the DSSRM algorithm are designed to this task: BF_DSSRM uses the brute-force method to compute the disjunctive support of candidates, while O_DSSRM relies on the optimized method. Hereafter, we simply use "DSSRM" to indicate both versions when describing common features. For the latter representation, we used the MEP algorithm (Casali, Cicchetti, and Lakhal 2005) whose source code was kindly provided by its authors.

All experiments were carried out on a PC equipped with an Intel processor having as clock frequency 1.73GHz and 2GB of main memory, and running the Linux distribution Fedora Core 7 (with 2GB of swap memory). DSSRM and MEP are implemented using the C++ language. The compiler used to generate executable codes is gcc 4.1.2. The performances of both versions of DSSRM and MEP are depicted by Figure 1. The obtained results point out the following assertions:

1. Efficient extraction of $\mathcal{DSSR}$: Indeed, except for very low values of *minsupp* (*e.g.*, *minsupp* = **20%** for PUMSB), all versions of DSSRM allow the extraction of the representation $\mathcal{DSSR}$ in a short time for all considered datasets. For very low values of *minsupp*, DSSRM versions also ensure an interesting mining time even in highly correlated datasets such as PUMSB.

2. Excessive extraction cost of the frequent essential pattern-based representation: Even for high values of *minsupp*, the mining time of the MEP algorithm is huge (*cf.* Figure 1). This can be easily noticed out from the figure associated to the ACCIDENTS dataset in which the runtime of MEP reaches its maximal value. This excessive mining time is mainly caused by two major factors, namely the use of an external algorithm to extract maximal frequent patterns, and the extraction of frequent essential patterns through their classical definition (*cf.* Definition 4, page 2). Indeed, MEP is obliged to uselessly compute the disjunctive supports of many candidates, which will be proved not to be essential patterns.

3. O_DSSRM is faster than BF_DSSRM: Thanks to O-METHOD, O_DSSRM is always faster than BF_DSSRM for all considered datasets (*cf.* Figure 1). Indeed, this method dramatically reduces the number of node visits required to compute the disjunctive supports of candidates. Such a reduction is clearly shown by Table 4, through **Ratio 1** depicted by the second and the third columns,

[4]O-METHOD stands for Optimized Method.

[5]BF-METHOD stands for Brute-Force Method.

[6]Test datasets are available at: *http://fimi.cs.helsinki.fi/data.*

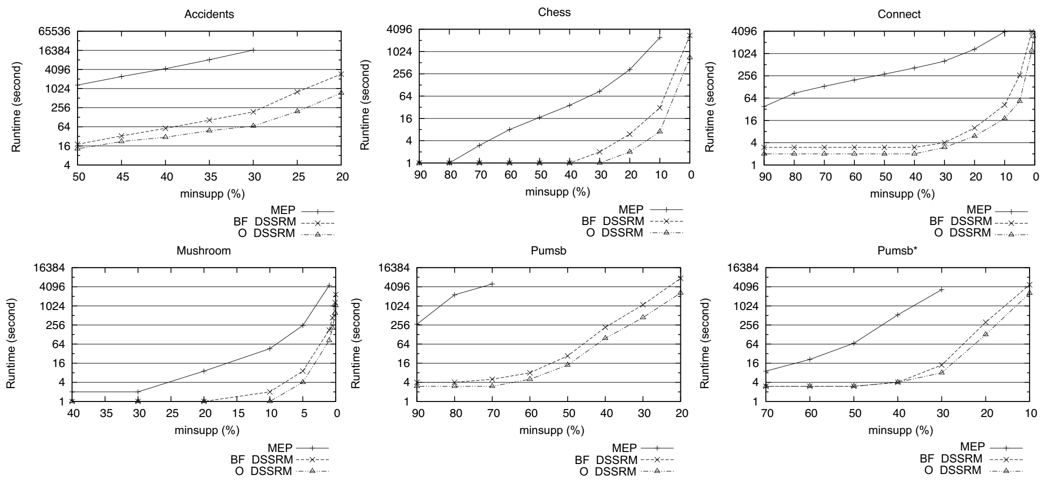

Figure 1: Performances of both versions of DSSRM *vs.* those of MEP.

	Ratio 1		Ratio 2	
Dataset	**Minimum**	**Maximum**	**Minimum**	**Maximum**
ACCIDENTS	**7.90** (50)	**16.62** (35)	**101.38** (50)	**245.10** (30)
CHESS	**2.77** (90)	**12.23** (10)	**1.00** (90)	**348.43** (10)
CONNECT	**4.03** (90)	**6.98** (40)	**19.00** (90)	**223.50** (20)
MUSHROOM	**1.99** (40)	**3.17** (1)	**2.00** (40)	**61.75** (5)
PUMSB	**6.28** (90)	**26.44** (50)	**90.00** (90)	**1, 674.00** (70)
PUMSB*	**1.63** (70)	**5.03** (30)	**3.00** (70)	**416.37** (30)

Table 4: Minimum and maximum values of Ratio 1 and Ratio 2 (*minsupp* values are given in % between brackets).

which respectively detail the minimum and the maximum values of the ratio comparing the number of visited nodes, respectively, by BF_DSSRM and O_DSSRM, *i.e.*, $\frac{\text{number of visited nodes using BF-METHOD}}{\text{number of visited nodes using O-METHOD}}$. Note that the ratio values are always greater than **1.63**.

4. All DSSRM versions are faster than MEP: For all tested datasets and especially for PUMSB and PUMSB*, the runtime of BF_DSSRM and O_DSSRM is drastically lower than that of MEP (*cf.* Figure 1). This mainly results from the characterization of essential patterns using their disjunctive closures. Indeed, contrary to MEP, both versions of DSSRM exploit the disjunctive closures and Proposition 9 to avoid computing the disjunctive supports of non-essential patterns by simply pruning them. The number of these patterns considerably increases as far as we decrease the *minsupp* value. Table 4 shows the minimum and maximum values of the ratio $\frac{\text{runtime of MEP}}{\text{runtime of O\_DSSRM}}$ for all considered datasets, through **Ratio 2** shown by the fourth and fifth columns. It is worth noting that, for each dataset, the ratio values increase when the *minsupp* values decrease, reaching a maximal value equal to **1, 674.00** times. This clearly shows that DSSRM is faster than MEP by several orders of magnitude for all datasets and even for low *minsupp* values.

Conclusion and future works

In this paper, we presented a thorough analysis of the new $\mathcal{DSSR}$ representation of frequent patterns defined through disjunctive patterns. Then, we introduced the DSSRM algorithm, allowing the extraction of the proposed representation. We also presented a study of the DSSRM properties as well as several designed optimization methods for enhancing its performances. Experiments showed that DSSRM allows an efficient extraction of the $\mathcal{DSSR}$ representation and that both its versions outperform MEP.

Our future work mainly address the design, the implementation and the evaluation of an algorithm allowing the regeneration of frequent patterns starting from the representation based on disjunctive closed patterns. In this respect, a preliminary study we carried out proved that the regeneration process is ensured to be efficient in comparison to the majority of the state of the art representations. Indeed, evaluating the conjunctive support of a pattern I requires at most the evaluation of a *unique* inclusion-exclusion identity starting from the representation based on disjunctive closed patterns. While, $2^{|I|}$ deduction rules are necessary, for example, in the case of (closed) non-derivable patterns (Hamrouni, Ben Yahia, and Mephu Nguifo 2009).

References

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference VLDB, Santiago, Chile*, 478–499.

Casali, A.; Cicchetti, R.; and Lakhal, L. 2005. Essential patterns: A perfect cover of frequent patterns. In *Proceedings of the 7th International Conference DaWaK, LNCS, volume 3589, Springer-Verlag, Copenhagen, Denmark*, 428–437.

Galambos, J., and Simonelli, I. 2000. *Bonferroni-type inequalities with applications*. Springer.

Hamrouni, T.; Ben Yahia, S.; and Mephu Nguifo, E. 2009. Sweeping the disjunctive search space towards mining new exact concise representations of frequent itemsets. *Data & Knowledge Engineering* 68(10):1091–1111.

Knuth, D. E. 1968. *The art of computer programming, volume 3*. Addison-Wesley.

Kryszkiewicz, M. 2009. Compressed disjunction-free pattern representation versus essential pattern representation. In *Proceedings of the 10th International Conference IDEAL, LNCS, volume 5788, Springer-Verlag, Burgos, Spain*, 350–358.

Smith, P. 2004. *Applied Data Structures with C++*. Jones & Bartlett.