Skypattern Mining:
from Pattern Condensed Representations to
Dynamic Constraint Satisfaction Problems

Willy Ugarte[1], Patrice Boizumault[1], Bruno Crémilleux[1], Alban Lepailleur[2],
Samir Loudni[1], Marc Plantevit[3], Chedy Raïssi[4] and Arnaud Soulet[5]

[1] *GREYC (CNRS UMR 6072) – University of Caen, F-14032 Caen - France*
[2] *CERMN (UPRES EA 4258 - FR CNRS 3038 INC3M) – University of Caen*
F-14032 Caen - France
[3] *Université de Lyon, CNRS, Université Lyon 1, LIRIS (UMR5205), F-69622 - France*
[4]*INRIA Nancy Grand-Est - France*
[5] *LI (EA 2101) – Université François Rabelais de Tours, F-41029 Blois – France*

Abstract

Data mining is the study of how to extract information from data and express it as useful knowledge. One of its most important subfields, pattern mining, involves searching and enumerating interesting patterns in data. Various aspects of pattern mining are studied in the theory of computation and statistics. In the last decade, the pattern mining community has witnessed a sharp shift from efficiency-based approaches to methods which can extract more meaningful patterns. Recently, new methods adapting results from studies of economic efficiency and multi-criteria decision analyses such as Pareto efficiency, or skylines, have been studied. Within pattern mining, this novel line of research allows the easy expression of preferences according to a dominance relation. This approach is useful from a user-preference point of view and tends to promote the use of pattern mining algorithms for non-experts. We present a significant extension of our previous work [1, 2] on the discovery of skyline patterns (or *"skypatterns"*) based on the theoretical relationships with condensed representations of patterns. We show how these relationships facilitate the computation of skypatterns and we exploit them to propose a flexible and efficient approach to mine skypatterns using a dynamic constraint satisfaction problems (CSP) framework.

We present a unified methodology of our different approaches towards the same goal. This work is supported by an extensive experimental study allowing us to illustrate the strengths and weaknesses of each approach.

Keywords: Skypatterns, pattern mining, constraint programming, dynamic CSP, user preferences.

1. Introduction

The process of extracting useful patterns from data, called *pattern mining*, is an important subfield of data mining, and has been used in a wide range of applications and domains such as bioinformatics [3], chemoinformatics [4], social network analysis [5], web mining [6] and network intrusion detection [7]. Since the key papers of Agrawal *et al.* [8], Mannila *et al.* [9], a considerable number of patterns, such as itemsets, strings, sequences, trees and graphs, have been studied and used in real-world applications. Nowadays, many pattern extraction problems like subgroup discovery [10], discriminative pattern mining [11], and tiling [12] are understood from both theoretical and computational perspectives.

Most existing pattern mining approaches enumerate patterns with respect to a given set of constraints that range from simple to complex. For instance, given a transaction database, a well-known pattern mining task is to enumerate all itemsets (i.e. sets of items) that appear in at least s transactions. However, the output of pattern mining operations can be extremely large even for moderately sized datasets. For instance, in the worst case, the number of frequent itemsets is exponential in the number of the items in the dataset.

So far, the community has expended much effort on developing sophisticated algorithms which push the constraints deep into the mining process [13]. But also in on compression (i.e. reduction) techniques to limit the number of output patterns depending on the application contexts [14, 15, 16]. The pattern mining community, however, has paid less attention to combining mining constraints. In practice, many constraints entail choosing threshold values such as the well-used minimal frequency. This notion of *"thresholding"* has serious drawbacks. Unless specific domain knowledge is available, the choice is often arbitrary and may lead to a very large number of extracted patterns which can reduce the success of any subsequent data analysis. This drawback is even more pronounced when several thresholds have to be combined. A second drawback is the *stringent enumeration aspect*: a pattern is either above or below a threshold. But what about patterns that respect only some thresholds? Should they be discarded? It is often very difficult to apply *subtle selection* mechanisms. There are very few works such as [17, 18] which propose to introduce a softness criterion into the mining process. Other studies attempt to integrate user preferences into the mining task in order to limit the number of extracted patterns such as the *top-k* pattern mining approaches [19, 20]. By associating each pattern with a *rank score*, this approach returns an ordered list of the k patterns with the highest score to the user. However, combining several measures in a single scoring function is difficult and the performance of top-k approaches is often sensitive to the size of the datasets and to the threshold value, k.

We present a unified methodology of two approaches that aim to make the results of pattern mining *useful from a user-preference point of view*. To this end, we integrate into the pattern discovery process the idea of skyline queries [21] in order to mine *skyline patterns* in a threshold-free manner. Such queries have attracted considerable attention due to their importance in multi-criteria decision making and economics where they are usually called *"Pareto efficiency or optimality queries"*. Briefly, in a multidimensional space where a preference is defined for each dimension, a point a dominates another point b if a is better (i.e. more preferred) than b in at least one dimension, and a is not worse than b on every other dimension. For example, a user selecting a set of patterns may prefer a pattern with a high frequency, a large length and a high confidence. In this case, we say that pattern a *dominates* another pattern b if $a.frequency \geq b.frequency$, $a.length \geq b.length$, $a.confidence \geq b.confidence$, where at least one strict inequality holds. Given a set of patterns, the skyline set contains the patterns that are not dominated by any other pattern.

Skyline pattern mining is interesting for several reasons. First, skyline processing does not require any threshold selection. In addition, for many pattern mining applications it is often difficult (or impossible) to find a reasonable global ranking function. Thus the idea of finding all optimal solutions in the pattern space with respect to multiple preferences is appealing. Second, the formal property of dominance satisfied by the skyline pattern defines a global interestingness measure with semantics easily understood by the user. These semantics are discussed at length in the economics literature, where the Pareto efficiency is applied to the selection of alternatives in resource distributions. However, while this notion of skylines has been extensively developed in engineering and database applications, it has remained unused for data mining purposes until recently [1]. Thirdly, skyline pattern mining is appealing from an efficiency and usability point of view. The authors of [22] established a loose upper-bound on the the average number of skyline tuples $O((\ln n)^{d-1})$ (with n tuples and d dimensions) which contrasts with the usual worst-case number of possible itemsets $O(2^{|\mathcal{I}|})$ (where $|\mathcal{I}|$ represents the cardinality of the set of items).

Contributions and roadmap

We present significant extensions of our recent papers [1, 2] on the discovery of skyline patterns, or *"skypatterns"*. First, we detail a *static* method (called Aetheris) based on the theoretical relationships with condensed representations of patterns (representations which return a subset of the patterns having the same expressiveness as the whole set of patterns [23]). Second, we describe a *dynamic* method (called CP+Sky) which involves a continuous refinement of the skyline constraints based on the extracted patterns. This is achieved through a dynamic CSP (Constraint Satisfaction Problems) framework (denoted by DynCSP). Third, the key notion of *"skylineability"* which constitutes the cornerstone of our two methods is explained in more detail. Finally, we present an extensive empirical study which includes a wide range of datasets and comparisons of our techniques. This enables us to draw some lessons about the strengths and weaknesses of each method and to better understand the advantages/weaknesses of the CSP machinery (see Sections 7.1.2 and 7.1.3).

The rest of this paper is organized as follows. Section 2 surveys the works related to skyline pattern analysis. Section 3 introduces some basic definitions, the formal problem statement and an overview of our work. The key notion of skylineability is then studied in Section 4. Section 5 discusses the computation of condensed representation of patterns for skypattern queries. Section 6 discusses skylineability but within a DynCSP framework. We report an empirical study on several datasets and a case study from the chemoinformatics domain in Section 7. Finally, Section 8 discusses the learnt lessons.

2. Related work

2.1. Pattern mining

Frequent itemset mining was first described in [8]. The problem can be defined as follows: a transaction is a subset of a given set of items $\mathcal{I}$, and a transaction database, denoted $\mathcal{T}$, is a set of such transactions. A subset x of $\mathcal{I}$ is a frequent itemset in $\mathcal{T}$ if the number of transactions containing x exceeds a given threshold, denoted by σ. One of the earliest findings in the data mining literature was that a mining process usually produces large collections of patterns. Many researchers have proposed methods to reduce the size of the output. These include the constraint-based pattern mining framework [24], the condensed representations [23] and the compression of the dataset by exploiting Minimum Description Length Principle [25], to name a few. A general observation is that patterns represent *"fragmented knowledge"*, and often there is no clear view of how these knowledge fragments interact and combine to produce a global model. Recent approaches have therefore used schemes such as pattern teams [26], constraint-based pattern set mining [27] and pattern selection strategies [28] that aim to minimize the redundancy and the number of patterns. A common theme in these studies is to select patterns from the initial large set of patterns on the basis of their usefulness in a given context. This approach falls into the general trend to produce *pattern sets* i.e. sets of patterns satisfying properties on the whole set of patterns [27]. Other approaches take advantage of *closed* patterns to maximize a specific measure such as the growth rate for emerging patterns [29] and the area for tiling [30, 12]. Often, these methods focus on optimizing a global measure on the discovered pattern set and neglect the relationships between patterns. Moreover, these approaches suffer from a lack of flexibility to express the queries requested by the analyst. For each method, the user has to understand its semantics and express queries satisfying its algorithmic properties and constraints.

Another class of techniques considers statistical significance of patterns. The objective is to extract patterns for which a given characteristic (usually the frequency) deviates so much from its expected value under a null model that it is unlikely to have been generated by it. The frequency of a pattern is then considered as a random variable, whose distribution under the null hypothesis has to be calculated or approximated, and the significance of the pattern is assessed through a statistical test that compares the expected frequency under the null model to the observed frequency. A number of works have explored various notions of statistical *significance* for itemsets and have proposed novel and efficient methods for their extraction [31, 32, 33, 34].

Pattern mining and Constraint programming. Pattern mining benefits from the recent cross-fertilization between data mining and Constraint Programming [35, 36, 37, 18]. Constraint Programming is a general declarative methodology for solving constraint satisfaction problems. Within this framework, the user specifies in a declarative way what the problem is by using constraints rather than a method dedicated to solve the problem. Then a general solver provides the complete set of solutions satisfying all the constraints. The approach is very expressive and allows to combine a wide range of mining constraints [36].

2.2. Skyline

The skyline points can be viewed as *compromise points* with respect to a given set of criteria. Skyline computation is strongly related to mathematical and microeconomics problems such as maximum vectors [38], Pareto set [39], and multi-objective optimization [40]. Since its rediscovery within the database community by Börzsönyi *et al.* [21], many methods have been developed for answering skyline queries that can handle various constraints in different computational environments [41, 42]. Skyline queries focus on the extraction of tuples from a given dataset and assume that all the elements are in the dataset, while the skypattern mining task consists of extracting patterns which are elements of the frontier defined by the given measures. The skypattern problem is clearly harder because the search space for skypatterns is much larger than the search space for skylines (cf. Section 3.2). Few studies focus on skypattern mining for several pattern domains (e.g., graphs and subgroups). The published approaches are designed for particular types of patterns and consider a very limited number of measures to compute the skyline of patterns. Among them, two proposals address graph analysis. In [43], the authors compute the skyline of subgraphs according to the number of vertices and the edge connectivity. Similarly, in [44], the authors adapt the framework of the *"Subdue"* method [45] to compute the patterns that are dominant according to three measures (e.g., frequency, number of nodes and density). In [46], the authors introduce the skypattern mining problem in the context of subgroups. Their approach aims at discovering subgroups that maximize a quality measure and a diversity measure. The notion of *dominance* is at the core of the skyline processing. In [47], the notion of dominance is used to propose a novel algebra extending relational algebras towards pattern mining. It leads to a generic method for mining several kinds of patterns (including the skypatterns) according to a preorder associated to the dominance relation. The solving part in [47] is performed by using Constraint Programming with a principle similar as the technique used in our CP+Sky method (cf. Section 6.2). The key idea is to use constraints on the dominance relation, which are dynamically added during the mining process. These constraints avoid producing solutions dominated by the solutions already extracted. In [47], the dynamically added constraints ensure that a candidate solution (i) is not dominated according to the preorder corresponding to the algebra or (ii) is equivalent to a solution already found. This last condition is required since the Pareto-dominance is a strict and partial order whereas a preorder is a reflexive relation. In CP+Sky, the dynamically added constraints stem from the dominance relation (i.e. a candidate solution is not dominated by the previous solutions). Finally, [47] does not deal with the skylineability notion which is introduced in our work. As we will see, the skylineability allows to reduce the number of measures that have to be considered in the mining process and thus decreasing the runtime. Skylineability is associated to the theoretical relationships that we establish between the skymining problem and condensed representations of patterns. Another option for preference-based processing is the *top-k* procedure [19, 20]. A ranking function f_r is applied to patterns, and the k best patterns with the highest score with respect to f_r are returned. As previously mentioned, this approach suffers from some limitations. The choice of k is not trivial (i.e. the *horizon* problem). A low value may miss useful patterns and too high a value introduces redundancy within the produced patterns (i.e. highly similar patterns). This limitation is the main motivation for the notion of the *"most informative patterns"* (MIP) that were proposed in [48]. MIPs can be seen as patterns that *locally dominate* other patterns according to a scoring function. This approach shares a similar spirit to ours as it also limits the number of enumerated patterns to a more manageable level. However, in contrast to our approach, work on MIPs includes a notion of dominance that is *local and specific* only to *subsets* of patterns.

3. Problem Statement and overview of the unified methodology

We introduce in this section some basic definitions and the formal problem statement. We also give an overview of the two methods Aetheris and CP+Sky we propose. These methods fully exploit an adequate representation of patterns dedicated to user-preferences [49]. Our study is interesting for several reasons. By carefully selecting patterns that are *"the best available"* for a given set of preferences, we greatly reduce the output and we limit the curse of *"pattern explosion"*. The user is *guaranteed* that only the best patterns w.r.t. his criteria are present in the final result. Last but not least, our approach is *threshold-free*[1]. Only the preferences and the dataset are required as an input.

3.1. Preliminary definitions

Although the problem can be formulated for any kind of pattern, for the sake of simplicity, we will illustrate our definitions using the itemset pattern domain. Section 8 discusses the computational and theoretical aspects associated with the problem when extracting more sophisticated kinds of patterns.

Let $\mathcal{I}$ be a set of distinct literals called *items*, an itemset (or pattern) corresponds to a non-empty subset of $\mathcal{I}$. These patterns are gathered together in the language $\mathcal{L}$: $\mathcal{L} = 2^{\mathcal{I}} \backslash \emptyset$. A transactional dataset $\mathcal{T}$ is a multi-set of patterns of $\mathcal{L}$. Each element of $\mathcal{T}$, named *transaction*, is a database entry. Table 1a presents a transactional dataset $\mathcal{T}$ where 6 transactions denoted by $t_1, \ldots, t_6$ are described by 6 items denoted by $A, \ldots, F$.

[1] Thresholds are entirely optional, depending on the analyst's needs and do not depend on the algorithm.

Tid	Items					
t_1	A	B	C	D	E	F
t_2	A	B	C	D	E	F
t_3	A	B				
t_4				D		
t_5	A		C			
t_6					E	

Items	val
A	10
B	55
C	70
D	30
E	15
F	25

(a) A toy data set $\mathcal{T}$

$Name$	Definition
$area$	$x \mapsto freq(x) \times length(x)$
$mean$	$x \mapsto \frac{min(x.val)+max(x.val))}{2}$
$bond$	$x \mapsto \frac{freq(x)}{freq_{\vee}(x)}$
$aconf$	$x \mapsto \frac{freq(x)}{max(x.freq)}$
gr_1	$x \mapsto \frac{\vert\mathcal{T}_2\vert}{\vert\mathcal{T}_1\vert} \times \frac{freq(x,\mathcal{T}_1)}{freq(x,\mathcal{T}_2)}$
$p\text{-}value$	$x \mapsto -binomial(prod(x.supp), freq(x))$

(b) Some measures of $\mathcal{M}$

Table 1: Example of a toy dataset and measures

All the measures discussed in this study are based on the set of *primitive-based measures* $\mathcal{M}$ that were defined in the context of constraint-based pattern mining [50]. Table 2 presents some general definitions of measures and Table 1b gives some specific examples (gr denotes the growth rate [11], $freq_{\vee}$ the disjunctive support, measures such as $bond$ and $aconf$ are detailed in [51]). Interestingly, our methodology is suitable for recent mining techniques utilizing statistical significance of patterns as discussed in related work. For instance, the p-value under the null model which considers all items to be independent random variables is rewritable as a primitive-based measure (see the definition of the $p\text{-}value$ in Table 1a). As claimed in [50], $\mathcal{M}$ encompasses a very large set of interesting measures.

In addition to the classical operators of $\Re^+$ (i.e. $+, -, \times, /$) and $\mathcal{L}$ (i.e. $\cup, \cap, \setminus$), the function $freq$ denotes the frequency of a pattern (i.e. $freq(x, \mathcal{T}) = |\{t \in \mathcal{T} \mid x \subseteq t\}|$), and $length$ its cardinality. The disjunctive support is $freq_{\vee}(x) = |\{t \in \mathcal{T} \mid \exists i \in x : i \in t\}|$. More atypical primitives also fit the primitive-based framework like $binomial(p, i) = \sum_{k=i}^{n} \binom{n}{k} p^k (1-p)^{n-k}$.

Given a function $val : \mathcal{I} \to \mathbb{R}^+$, we extend it to a pattern x and denote by $x.val$ the multi-set $\{val(i) \mid i \in x\}$. This kind of function is used with the usual SQL-like primitives sum, min and max. For instance, $sum(x.val)$ is the sum of val for each item of x. Note that $prod$ is a slightly different aggregate function due

Measure $m \in \mathcal{M}$	Primitive(s)	Operand(s)
$m_1 \theta m_2$	$\theta \in \{+, -, \times, /, binomial\}$	$(m_1, m_2) \in \mathcal{M}^2$
$\theta(s)$	$\theta \in \{freq, freq_{\vee}, length\}$	$s \in \mathcal{S}$
$\theta(s.val)$	$\theta \in \{sum, max, min, prod\}$	$s \in \mathcal{S}$
constant $r \in \Re^+$	-	-

Syntactic expression $s \in \mathcal{S}$	Primitive(s)	Operand(s)
$s_1 \theta s_2$	$\theta \in \{\cup, \cap, \setminus\}$	$(s_1, s_2) \in \mathcal{S}^2$
$\theta(s)$	$\theta \in \{f, g\}$	$s \in \mathcal{S}$
variable $x \in \mathcal{L}$	-	-
constant $l \in \mathcal{L}$	-	-

Table 2: A subset of the primitive-based measures

to $val : \mathcal{I} \to [0, 1]$ (e.g., the support of each item in p-value definition). Finally, f is the intension i.e. $f(T) = \{i \in \mathcal{I} \mid \forall t \in T, \ i \in t\}$, and g is the extension i.e. $g(x) = \{t \in Tid \mid x \subseteq t\}$.

This large variety of measures allows for more flexibility to formulate new or well-known interestingness measures that match the data analyst's objectives. Rather than using a ranking function for combining these measures and then maximizing it, we propose to use the Pareto composition:

Definition 1 ((Pareto-)Dominance). *Given a set of measures $M \subseteq \mathcal{M}$, a pattern x* dominates *another pattern y with respect to M, denoted by x $\succ_M$ y, iff for any measure $m \in M$, $m(x) \geq m(y)$ and there exists $m \in M$ such that $m(x) > m(y)$. Two patterns x and y are said to be* indistinct *with respect to M, denoted by x $=_M$ y, iff $m(x)$ equals $m(y)$ for any measure $m \in M$ (if $M = \emptyset$, then x $=_\emptyset$ y). Finally, x $\succeq_M$ y denotes that $(x \succ_M y) \vee (x =_M y)$.*

Note that we define the Pareto dominance only with the greater than symbol (i.e. $>$) assuming that the end-user wants to maximize a set of measures. The case of a minimization of a measure m is equivalent to maximizing the measure $m' = -m$ (this case is illustrated with the definition of $p\text{-}value$ which contains a minus).

Consider our running example using the data set $\mathcal{T}$ in Table 1a and suppose that $M = \{freq, area\}$, then the pattern $ABCDEF$ dominates ABC because $freq(ABC) = freq(ABCDEF) = 2$ and $area(ABCDEF) > area(ABC)$. Note in this case that $ABCDEF$ is indistinct to ABC with respect to $\{freq\}$. Similarly, suppose that $M = \{freq, mean, length\}$, the pattern AC dominates

AB because $freq(AC) = freq(AB) = 3$, $|AB| = |AC| = 2$ and $mean(AC) > mean(AB)$.

3.2. The skypattern mining problem

Given a set of measures M, if a pattern is dominated by another according to all measures of M, it is irrelevant and should be discarded in the output. The notion of *skyline pattern*, skypattern for short, formalizes this intuition.

Definition 2 (Skypattern operator). *Given a pattern set $P \subseteq \mathcal{L}$ and a set of measures $M \subseteq \mathcal{M}$, a skypattern of P with respect to M is a pattern not dominated by any pattern in P with respect to M. The skypattern operator $Sky(P, M)$ returns all the skypatterns of P with respect to M:*

$$Sky(P, M) = \{x \in P \mid \nexists y \in P : y \succ_M x\}$$

Then, the skypattern problem can be stated:

Problem 1. *Given a set of measures $M \subseteq \mathcal{M}$, the* skypattern mining problem is to evaluate the query $Sky(\mathcal{L}, M)$.

For instance, from the running data set (cf. Table 1a), $Sky(\mathcal{L}, \{freq, length\}) = \{ABCDEF, AB, AC, A\}$, as illustrated in Figure 1.

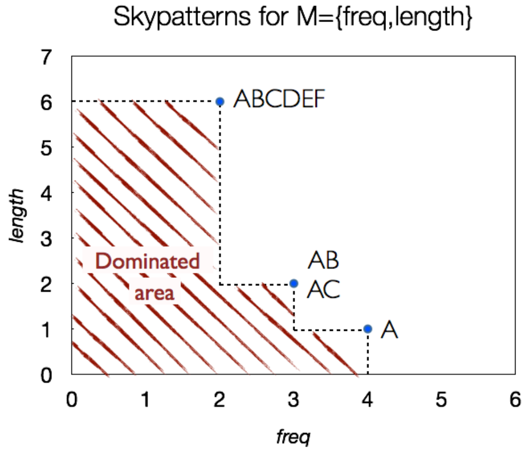


Figure 1: Example of skypatterns for the set of measures $M = \{freq, length\}$ (all the other patterns are in the dominated area).

In the general case, the skypattern mining problem is challenging because of the very high number of candidate patterns (i.e $|\mathcal{L}|$). Indeed, a naive enumeration of $\mathcal{L}$ is not feasible. For example, with 1000 items a naive approach will need to compute $(2^{1000} - 1) \times |M|$ measures and then compare them. A less naive approach based on heuristics (such as the anti-monotonicity of some measures) may give some results. However, the performance will be closely tied to the underlying properties of the data sets. For instance, in the case of the frequency measure, the density of the data set plays a major role in the performance and some algorithms are not able to extract frequent patterns at very low thresholds. Nevertheless, considering the following property provides new insights into an efficient computation of skypattern queries.

Property 1. *Given a set of measures $M \subseteq \mathcal{M}$, $\mathcal{S}ky(\mathcal{L}, M) = \mathcal{S}ky(P, M)$ for any pattern set P such that $\mathcal{S}ky(\mathcal{L}, M) \subseteq P$,*

$$(\forall P \subseteq \mathcal{L})(\mathcal{S}ky(\mathcal{L}, M) \subseteq P \Rightarrow \mathcal{S}ky(\mathcal{L}, M) = \mathcal{S}ky(P, M))$$

Proof Let P be a set of patterns such that $\mathcal{S}ky(\mathcal{L}, M) \subseteq P \subseteq \mathcal{L}$. First, let x be a pattern in $\mathcal{S}ky(\mathcal{L}, M)$. Then there is no $y \neq x$ in $\mathcal{L}$ such that $y \succ_M x$. In particular, there is no $y \neq x$ in P such that $y \succ_M x$. Note that x belongs to P which is a superset of $\mathcal{S}ky(\mathcal{L}, M)$. Thus, x is a pattern in $\mathcal{S}ky(P, M)$. Suppose now that x is not in $\mathcal{S}ky(\mathcal{L}, M)$: then there exists $y' \neq x$ in $\mathcal{L} \setminus P$ such that $y' \succ_M x$ and $\forall y'' \in P$, there is $y'' \not\succ_M x$. By induction, any y' or another pattern dominating y' not in P would have to be in $\mathcal{S}ky(\mathcal{L}, M)$.

As $\mathcal{S}ky(\mathcal{L}, M) \subseteq P \subseteq \mathcal{L}$ and $|P| \leq |\mathcal{L}|$, we argue that evaluating $\mathcal{S}ky(P, M)$ is generally much less costly than evaluating $\mathcal{S}ky(\mathcal{L}, M)$ since the cost of $\mathcal{S}ky(x, M)$ generally decreases with the cardinality of x. Consequently, we aim to reduce the cost of evaluating $\mathcal{S}ky(P, M)$ by finding a small but relevant set P (i.e. that includes $\mathcal{S}ky(\mathcal{L}, M)$) by means of 1) condensed representations of patterns or 2) dynamic pruning.

Condensed representations of patterns. In many pattern mining tasks (e.g., association rule mining or clustering), condensed representations of patterns significantly reduce the mining effort without loss of precision. Could we use this principle in the case of skypattern mining? A direct approach would be to compute a concise representation for each measure $m \in M$, but this is generally not possible because some measures, such as area or length, are *not condensable* (i.e., the condensed representation is equal to $|\mathcal{L}|$). Therefore, our problem can be reformulated as follows: *given a set of measures M, how to identify a smaller set of measures M' which allows the computation of a concise representation on the* *patterns (i.e. the pattern set P) without loss of skypatterns? In addition, how can one use this set of measures to extract efficiently the skypatterns without redundancies?* We address this problem in the next sections.

Dynamic pruning. Instead of extracting the whole condensed representation of patterns and then applying the $\mathcal{S}ky$ operator. One may consider the use of the $\mathcal{S}ky$ operator locally during the extraction. Indeed, as soon as a pattern is a candidate skypattern, the search space dominated by this pattern can be directly eliminated. In other words, each new candidate skypattern adds a constraint allowing to safely prune the remaining search space. These new constraints prevent the enumeration of unnecessary patterns. Section 6 describes how CSP can be used to update constraints during the extraction and thus reduce the search space.

3.3. Unified methodology for the two methods

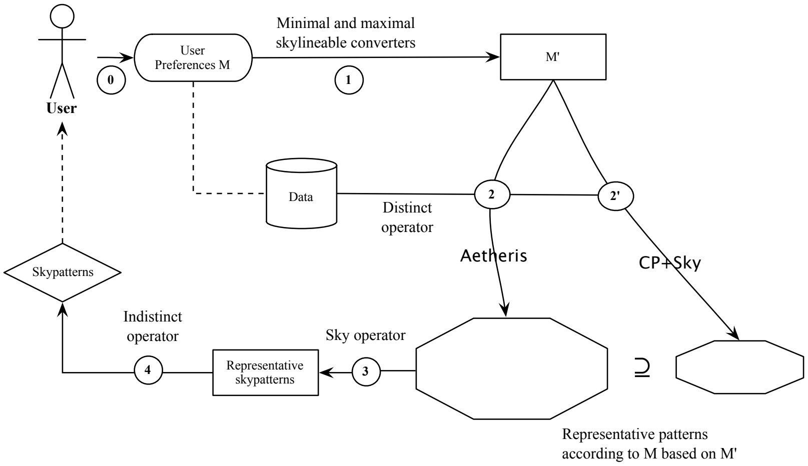

User
0
User Preferences M
Minimal and maximal skylineable converters
1
M'
Data
Distinct operator
2
2'
Aetheris
CP+Sky
Skypatterns
Indistinct operator
Sky operator
4
Representative skypatterns
3
⊇
Representative patterns according to M based on M'

Figure 2: Unified view of skypattern mining with Aetheris and CP+Sky.

To clarify our methodology, we illustrate in Figure 2 the different processes of the two methods Aetheris and CP+Sky we propose. These two methods share the same overall methodology and mainly differ in the specific step of the computation of representative patterns of the skypatterns.

After the user's preferences selection, in a common first step, Aetheris and CP+Sky automatically identify a smaller set of measures M' which allows for the computation of a concise representation on the patterns thanks to the use of *converters* (cf. Section 4). The second step (respectively 2 and 2' for Aetheris and CP+Sky) aims at computing an as small as possible superset that enables the retrieval of all skypatterns. For that purpose, Aetheris builds a static set of representative patterns according to the set M', which is based on the notion of *converting* the initial set of preference M (cf. Section 5). CP+Sky builds dynamically a more concise set by pruning unpromising patterns (the set of representative patterns of CP+Sky is included in the set of representative patterns of Aetheris) (cf. Section 6). The third step filters the set of representative patterns with the Sky operator. This step remains efficient because the number of representative patterns is much smaller than the number of possible skypatterns. Finally, this step provides a concise representation of the skypatterns. The end-user can either output this concise representation or the entire list of skypatterns as a final step depending on the application needs. Our methodology revolves around the simple idea that to be able to efficiently extract and analyze skypatterns, one needs to be able to (statically or dynamically) exhibit a concise representation of the skypatterns that will be used as an input to the skyline operator.

4. Skylineability

The set of skypatterns has no good property like downward closure or convexity. Conventional techniques used to prune the pattern search space like anti-monotone properties are thus ineffective. However, using Property 1, an efficient computation of the set of representative patterns of skypatterns becomes possible. To do so, we introduce the notion of skylineability which aims at computing a reduced collection of representative patterns of the skypatterns (cf. the end of the previous section). This efficient computation is carried out using either free [16] or closed patterns [14].

4.1. Skylineability of a set of measures

Intuitively, skylineability refers to the notion of local extrema in the search space. The local extrema in this case are the maximal values for each preference selected by the end-user. By definition, only these extrema may therefore be skypatterns. Thus there is no need, while mining, to take into account the other patterns which are necessarily dominated by these extrema. Assume any two patterns x and y, such that $x \subset y$, have the same value for each measure of M': $x =_{M'} y$. If y always dominates x for a given set of measures $M \subseteq \mathcal{M}$, then M is said to be maximally M'-skylineable. This information is extremely important as it allows the discarding of non-maximal patterns (here, x) which are dominated by maximal patterns (here, *y*). This notion of skylineability can be seen as a way of *mapping domination* between two different sets of measures M and M'.

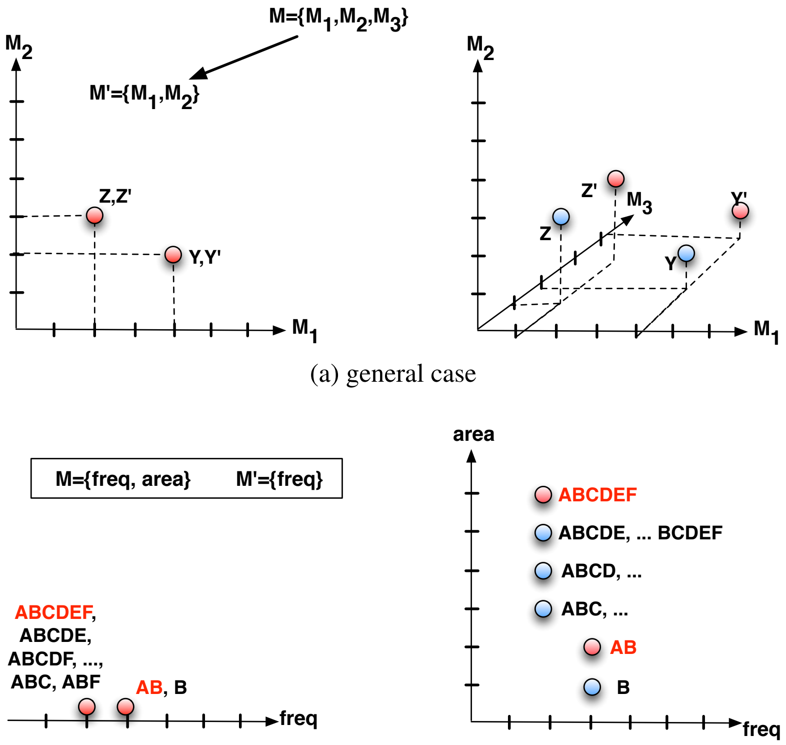
Figure 3 (top) depicts the benefit of skylineability in the general case where it becomes possible to focus only on a subset of patterns and measures (i.e. M') and have a formal guarantee that these patterns will not be dominated in the full set of measures (i.e. M). We illustrate now this notion of skylineability on our running example. Let us consider patterns from $\mathcal{T}$ that maximize the frequency and area measures: $M = \{\mathit{freq}, area\}$. In our example (cf. Table 1a), the patterns B and AB have the same frequency (see Figure 3 (bottom)). Thus, if we define $M' = \{freq\}$, $B =_{M'} AB$ and $AB \succ_M B$ because the area of AB is greater than that of B. In fact, for any two patterns $x \subset y$ such that $x =_{\mathit{freq}} y$, we have $y \succ_M x$ and $M = \{\mathit{freq}, area\}$ is thus said *maximally* $\{\mathit{freq}\}$*-skylineable*. The mining process can focus on only extracting patterns based on the frequency measure (i.e. M') without having to take into account the area measure that is present in M.

In practice, local extrema are not necessarily the longest patterns (i.e. the closed patterns which are the maximal patterns of equivalence classes) but can also be the shortest patterns (i.e. the free patterns which are the minimal patterns of equivalence classes). Thus, the concept of skylineability is defined in a dual manner:

Definition 3 (Minimal skylineability). *Given a set of measures $M' \subseteq \mathcal{M}$, a set of measures $M \subseteq \mathcal{M}$ is said to be (strictly) minimally M'-skylineable iff for any patterns x and y such that $x \subset y$ and $x =_{M'} y$, one has $x \succeq_M y$ (respectively $x \succ_M y$).*

Definition 4 (Maximal skylineability). *Given a set of measures $M' \subseteq \mathcal{M}$, a set of measures $M \subseteq \mathcal{M}$ is said to be (strictly) maximally M'-skylineable iff for any patterns x and y such that $x \subset y$ and $x =_{M'} y$, one has $y \succeq_M x$ (respectively $y \succ_M x$).*

From the previous definitions, given a set of measures M which is maximally M'-skylineable, if $x =_{M'} y$ and $x \supset y$, it is clear that *x* cannot be dominated by *y* on M. For instance, $M = \{\mathit{freq}, area\}$ is strictly maximally $\{\mathit{freq}\}$-skylineable because $area(x)$ strictly increases with the cardinality of *x* (when the frequency remains constant). Hence, in Figure 3 (bottom), we can deduce that $ABCDEF$ dominates the patterns $ABCDE$, $ABCD$, $\ldots$, DEF without considering the full set of measures M but only M'. Notice that $\{\mathit{freq}\}$ is (weakly) maximally (or minimally) $\{\mathit{freq}\}$-skylineable and that $\{length(x)\}$ is strictly maximally $\emptyset$-skylineable.



(a) general case

(b) running example

Figure 3: Use of skylineability in the general case (cf. Figure 3a, top) and in our running example (cf. Figure 3b, bottom) in which $AB =_{freq} B$: we directly knows that $AB \succ_M B$ without considering $area$ ($length(AB) > length(B)$).

Property 2. *Given a set of measures $M \subseteq \mathcal{M}$, there is at least one set $M' \subseteq \mathcal{M}$ such that M is minimally and maximally M'-skylineable.*

Proof Let $M \subseteq \mathcal{M}$ be a set of measures. Let $M' \subseteq \mathcal{M}$ be the set of all unary primitives defined on $\mathcal{L}$. Let x and y be two patterns such that $x =_{M'} y$. Let $m \in M$. As $x =_{m'} y$ for any primitive $m' \in M'$ and m is composed of such primitives, thus $x =_m y$. We conclude that $x =_M y$ and then, $x \succeq_M y$ and $y \succeq_M x$.

Property 2 is a very important result as it means that *a set of measures is always skylineable* (due to the fact that M is at least M-skylineable). Obviously, for a set of measures M, the smaller[2] M', the stronger its M'-skylineability. For in-

[2] In the sense of cardinality.

stance, $\{freq\}$-skylineability is more interesting than $\{freq, area\}$-skylineability because $area$ is not a condensable function [49]: there is no pair of distinct patterns x and y such that $x =_{\{freq,area\}} y$. How to choose automatically a subset M' is discussed next.

4.2. Minimal and maximal skylineable converters

One of the disadvantages of skylineability is that it depends on a set of measures M' whose choice is essential to effectively reduce the search space. We propose two operators to automatically build M' given the initial set of measures M. Basically, the construction of M' consists in identifying primitives that must remain constant in order that minimal or maximal patterns are always dominant patterns. Intuitively, any primitive p that is part of the measure $m \in M$ that hinders the M'-skylineability of m, has to be added to M'. For instance, it is easy to see that the frequency decreases the area because the frequency decreases with the specialization. In order that the closed patterns maximize the area, the frequency has to belong to M'. More generally, it is essential to take into account the monotone behavior of primitives: decreasing or increasing. For instance, the length increases with x while the frequency decreases. It is also necessary to consider the operations that combine these primitives. The result of a multiplication increases when one of its operands increases. However, the result of a division decreases with its second operand. For this purpose, we define two operators denoted $\underline{c}$ and $\overline{c}$ (see Table 3).

Expr. e	Primitive(s)	$\underline{c}(e)$	$\overline{c}(e)$
$e_1 \theta e_2$	$\theta \in \{+, \times, \cup\}$	$\underline{c}(e_1) \cup \underline{c}(e_2)$	$\overline{c}(e_1) \cup \overline{c}(e_2)$
$e_1 \theta e_2$	$\theta \in \{-, /, binomial, \cap\}$	$\underline{c}(e_1) \cup \overline{c}(e_2)$	$\overline{c}(e_1) \cup \underline{c}(e_2)$
constant	-	$\emptyset$	$\emptyset$
$d(x)$	$d \in \{freq, min, g, prod\}$	$\emptyset$	$\{d(x)\}$
$i(x)$	$i \in \{length, max, sum, freq_\vee, f\}$	$\{i(x)\}$	$\emptyset$
$d(e_1)$	$d \in \{freq, min, g, prod\}$	$\overline{c}(e_1)$	$\underline{c}(e_1)$
$i(e_1)$	$i \in \{length, max, sum, freq_\vee, f\}$	$\underline{c}(e_1)$	$\overline{c}(e_1)$

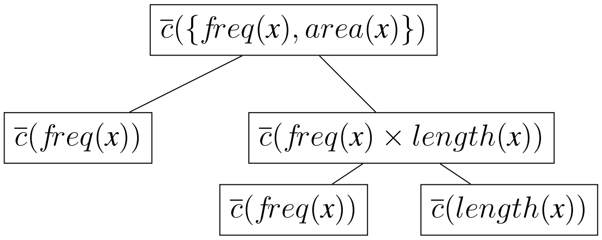
Table 3: Definition of the minimal and maximal skylineable converters $\underline{c}$ and $\overline{c}$

Given a primitive-based measure $m \in \mathcal{M}$, the minimal skylineable converter returns a set of measures $M' = \underline{c}(m)$ guaranteeing that for any pattern $x \subset y$, if $x =_{M'} y$ then $m(x) \geq m(y)$. In other words, x dominates y with respect to m. Dually, the maximal converter $\overline{c}$ guarantees that $m(x) \leq m(y)$ for any pattern $x \subset y$ such that $x =_{\overline{c}(m)} y$.

Let us illustrate $\underline{c}$ and $\overline{c}$ on the area measure. The area is defined as a product of the frequency and length. Thus, we use the first definition in Table 3. $\underline{c}(area) = \underline{c}(\mathit{freq}(x)) \cup \underline{c}(length(x)) = \emptyset \cup \{length(x)\} = \{length(x)\}$. Symmetrically, $\overline{c}(area) = \overline{c}(\mathit{freq}(x)) \cup \overline{c}(length(x)) = \{\mathit{freq}(x)\} \cup \emptyset = \{\mathit{freq}(x)\}$. The skylineable converters enable us to automatically find optimization techniques already known for specific measures such as area [30, 12], p-value [32] or growth-rate [29] (see Table 4a). We can observe that many measures are maximized by the closed itemsets according to frequency which may explain the success of closed pattern mining. But, in this work, we *generalize* this principle to optimize *any* primitive-based measures. Note that when the converter $\underline{c}$ returns no measure (e.g., $bond$ or $aconf$), it means that the measure decreases with respect to the specialization. Dually, $\overline{c}(m) = \emptyset$ means that m increases with respect to the specialization.

Meas. m	$\underline{c}(m)$	$\overline{c}(m)$
$area$	$\{length(x)\}$	$\{\mathit{freq}(x)\}$
$mean$	$\{max(x.val)\}$	$\{min(x.val)\}$
$bond$	$\emptyset$	$\{\mathit{freq}(x), \mathit{freq}_{\vee}(x)\}$
$aconf$	$\emptyset$	$\{\mathit{freq}(x), max(x.val)\}$
gr_1	$\{\mathit{freq}(x, \mathcal{T}_2)\}$	$\{\mathit{freq}(x, \mathcal{T}_1)\}$
$p\text{-}value$	$\{prod(x.supp)\}$	$\{\mathit{freq}(x)\}$

(a) Individual measures


$\overline{c}(\{\mathit{freq}(x), area(x)\})$
$\overline{c}(\mathit{freq}(x))$
$\overline{c}(\mathit{freq}(x) \times length(x))$
$\overline{c}(\mathit{freq}(x))$
$\overline{c}(length(x))$

(b) A set of measures $M = \{\mathit{freq}(x), area(x)\}$

Table 4: Applying the minimal and maximal converters

In practice, as the skypatterns are computed for a set of measures, we extend the minimal and maximal converters:

Definition 5 (Minimal and maximal skylineable converters). *The minimal and maximal skylineable converters defined by Table 3 for any primitive-based measure are naturally extended to a set of primitive-based measures* $M \subseteq \mathcal{M}$*:* $\overline{c}(M) = \bigcup_{m \in M} \overline{c}(m)$ *and* $\underline{c}(M) = \bigcup_{m \in M} \underline{c}(m)$*.*

For instance, $\overline{c}(\{\mathit{freq}(x), area(x)\}) = \overline{c}(\mathit{freq}(x)) \cup \overline{c}(area(x)) = \{\mathit{freq}(x)\}$ and $\underline{c}(\{\mathit{freq}(x), area(x)\}) = \underline{c}(\mathit{freq}(x)) \cup \underline{c}(area(x)) = \{length(x)\}$. $\overline{c}(\{\mathit{freq}(x), area(x)\}) = \{\mathit{freq}(x)\}$ means that the most specific patterns (when the frequency remains unchanged) maximize the measures $\{\mathit{freq}(x), area(x)\}$. The following property formalizes this observation:

Property 3. *A set of primitive-based measures $M \subseteq \mathcal{M}$ is minimally $\underline{c}(M)$-skylineable and maximally $\overline{c}(M)$-skylineable.*

Proof The key idea relies on the monotonous property according to each variable of a primitive. The operators $\underline{c}(\centerdot)$ and $\overline{c}(\centerdot)$ are recursively applied to return the set of primitives which must be constant in order to respectively minimize or maximize the measure. Two cases arise to be sure that an expression $p(h_1, \ldots, h_k)$ is minimized with x. For each $i \in \{1, \ldots, k\}$, if the primitive p increases according to the i^{th} variable (while the other ones remain unchanged), it is necessary to return the primitives such that h_i is also minimized according to x. For this purpose, the minimal skylineable converter is applied again. Otherwise, the primitive p decreases with the i^{th} variable and we return the primitives such that h_i is maximized by applying the maximal skylineable converter. The dual approach is used to maximize a measure.

In our implementation, the set of measures M is parsed through a syntax tree. Following this step, the minimal and maximal skylineable converters are recursively applied to automatically compute $\underline{c}(M)$ and $\overline{c}(M)$ (an example is provided in Table 4b for $M = \{\mathit{freq}(x), area(x)\}$). This process is illustrated in Figure 2 with the edge labelled 1. From now on, the set of measures M' refers to $\underline{c}(M)$ or $\overline{c}(M)$.

5. Condensed representations of patterns for static mining of skypatterns

This section presents our static method called Aetheris based on the theoretical relationships between condensed representations of patterns and skypatterns. Aetheris follows a two-step process (cf. Section 3.3): first, a set of representative patterns is extracted and then the $\mathcal{Sky}$ operator is applied on these patterns. The technique is said to be *static* because the extraction of a representative pattern does not depend on the patterns already extracted, contrary to the CP+Sky method presented in the next section.

A major issue is how to extract representative patterns for a group of skypatterns. In the previous section, we remarked that some skypatterns share exactly the same values on the whole set of measures M' (e.g. $B =_{\{freq\}} AB$). This observation enables us to properly answer the question: instead of directly evaluating the skypattern query on $\mathcal{L}$, we can compute the skypatterns on a condensed representation of $\mathcal{L}$ and then generate the entire set of skypatterns. To this end, we introduce the *distinct operator* which is at the core of the construction of a condensed representation adequate to M':

Definition 6 (Distinct operator). *Given a set of measures $M' \subseteq \mathcal{M}$, the distinct operator for $P \subseteq \mathcal{L}$ with respect to M' and $\theta \in \{\subset, \supset\}$ returns all the patterns x of P such that their generalizations (or specializations) are distinct from x with respect to M':*

$$\mathcal{D}is_{\theta}(P, M') = \{x \in P \mid \forall y \theta x : x \neq_{M'} y\}$$

where $\theta \in \{\subset, \supset\}$.

Given a set of measures M', the set of free (respectively closed) patterns adequate to M' corresponds exactly to $\mathcal{D}is_{\subset}(\mathcal{L}, M')$ (respectively $\mathcal{D}is_{\supset}(\mathcal{L}, M')$). For instance, from our running example, $\mathcal{D}is_{\subset}(\mathcal{L}, \{freq\}) = \{A, B, C, D, E, F, AD, AE, BC, BD, BE, CD, CE, DE\}$ and $\mathcal{D}is_{\supset}(\mathcal{L}, \{freq\}) = \{A, D, E, AB, AC, ABCDEF\}$.

We now introduce the *indistinct operator* that enables the retrieval of all the indistinct patterns from their representatives:

Definition 7 (Indistinct operator). *Given a set of measures $M' \subseteq \mathcal{M}$, the indistinct operator returns all the patterns of $\mathcal{L}$ being indistinct with respect to M' with at least one pattern in P.*

$$\mathcal{I}nd(\mathcal{L}, M', P) = \{x \in \mathcal{L} \mid \exists y \in P : x =_{M'} y\}$$

For instance, from Table 1a, the set of patterns that have exactly the same frequency as patterns B or C is $\mathcal{I}nd(\mathcal{L}, \{freq\}, \{AB, AC\}) = \{B, C, AB, AC\}$.

Preserving functions express a property of compression and are at the core of Property 4. A preserving function is a condensable primitive (many functions are preserving: $freq$, $freq_{\vee}$, $count$, min, max, sum, etc., see more details in [49]).

Property 4. *Given a set of preserving functions M', one has the following relation for any $P \subseteq \mathcal{L}$ and $\theta \in \{\subset, \supset\}$:*

$$\mathcal{I}nd(P, M', \mathcal{D}is_{\theta}(P, M')) = P$$

In other words, the indistinct operator is the inverse function for the distinct operator. For instance, $\mathcal{I}nd(\mathcal{L}, \{\mathit{freq}\}, \mathcal{D}is_{\supset}(\{B, C, AB, AC\}, \{\mathit{freq}\})) = \{B, C, AB, AC\}$.

These operators are the basis of an efficient technique to compute skypatterns. The key principle is to confront only distinct patterns together instead of individually comparing each pattern. Indeed, the computation of skypatterns with respect to $M = \{\mathit{freq}, area\}$ can be limited to $\mathcal{D}is_{\supset}(\mathcal{L}, \{\mathit{freq}\})$ because maximal $\{\mathit{freq}\}$-skylineability guarantees us that the other patterns are not dominant patterns. For instance, as $AB =_{\mathit{freq}} B$, the $\{\mathit{freq}\}$-skylineability of M gives $AB \succ_M B$ and B cannot be a skypattern. More formally, we know that $\mathcal{S}ky(\mathcal{I}nd(\mathcal{L}, M', \mathcal{D}is_{\theta}(\mathcal{L}, M')), M) = \mathcal{S}ky(\mathcal{L}, M)$ from Property 4. Theorem 1 now proves that the skypattern operator can be pushed into the indistinct operator:

Theorem 1 (Operational equivalence). *If a set of measures M is M'-skylineable with respect to $\theta \in \{\subset, \supset\}$ and M' is a set of measures, then one has:*

$$\mathcal{S}ky(\mathcal{L}, M) = \mathcal{I}nd(\mathcal{L}, M, \mathcal{S}ky(\mathcal{D}is_{\theta}(\mathcal{L}, M'), M))$$

Proof Let M be a set of measures M'-skylineable with $\theta \in \{\subset, \supset\}$.
1. $\mathcal{S}ky(\mathcal{L}, M) \supseteq \mathcal{I}nd(\mathcal{L}, M, \mathcal{S}ky(\mathcal{D}is_{\theta}(\mathcal{L}, M'), M))$.
Let $x \in \mathcal{I}nd(\mathcal{L}, M, \mathcal{S}ky(\mathcal{D}is_{\theta}(\mathcal{L}, M'), M))$ and $y \in \mathcal{L}$.
There exist $x' \in \mathcal{S}ky(\mathcal{D}is_{\theta}(\mathcal{L}, M'), M)$ such that $x' =_M x$ and $y' \in \mathcal{D}is_{\theta}(\mathcal{L}, M')$ such that $y' =_{M'} y$ and $y' \succeq_M y$ (i.e. M'-skylineability). As x' belongs to $\mathcal{S}ky(\mathcal{D}is_{\theta}(\mathcal{L}, M'), M)$, it cannot be dominated by any pattern of $\mathcal{D}is_{\theta}(\mathcal{L}, M')$: $y' \not\succ_M x$. Thus, x is not dominated by y (i.e. x is a skyline of $\mathcal{L}$ with respect to M) because $x' =_M x$ and $y' \succeq_M y$.
2. $\mathcal{S}ky(\mathcal{L}, M) \subseteq \mathcal{I}nd(\mathcal{L}, M, \mathcal{S}ky(\mathcal{D}is_{\theta}(\mathcal{L}, M'), M))$ such that $y' =_{M'} y$ and $y' \succeq_M y$. As y is a skypattern, one has $y \succeq_M x'$ and thus, $y' =_M y$. Furthermore, no pattern of $\mathcal{D}is_{\theta}(\mathcal{L}, M')$ dominates y nor y': $y' \in \mathcal{S}ky(\mathcal{D}is_{\theta}(\mathcal{L}, M'), M)$. Finally, as $y' =_M y$, y belongs to $\mathcal{I}nd(\mathcal{L}, M, \mathcal{S}ky(\mathcal{D}is_{\theta}(\mathcal{L}, M'), M))$.

The skypatterns of $\mathcal{S}ky(\mathcal{D}is_{\theta}(\mathcal{L}, M'), M)$ form a condensed representation of $\mathcal{S}ky(\mathcal{L}, M)$. It is well-known that the size of adequate condensed representations (i.e. $\mathcal{D}is_{\subset}(\mathcal{L}, M')$ or $\mathcal{D}is_{\supset}(\mathcal{L}, M')$) is smaller than the whole collection of patterns [23]. Thus, we have achieved our objective as mentioned in Section 3.2.

The technique is even more efficient if the set of measures is *strictly* M'-skylineable. In this case, the $\mathcal{I}nd$ operator can be skipped and Theorem 1 is

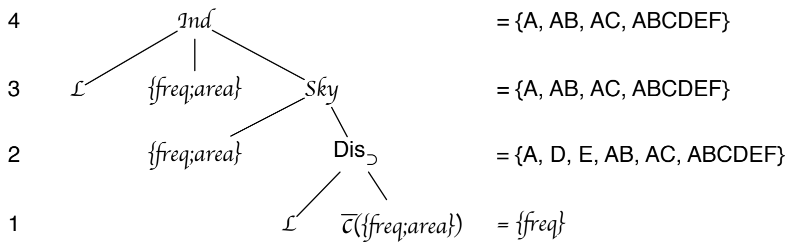


Figure 4: Computing the skypatterns with respect to $\{freq; area\}$ from the running example

reduced to the following relation: $\mathcal{S}ky(\mathcal{L}, M) = \mathcal{S}ky(\mathcal{D}is_\theta(\mathcal{L}, M'), M)$ (with $\theta \in \{\subset, \supset\}$).

Figure 4 illustrates the computation of the skypatterns with our method Aetheris. Suppose that $M = \{freq, area\}$, the first step applies the maximal skylineable converter on M. Then, the distinct operator preserves the closed itemsets (Step 2). The skyline operator selects the dominant patterns at Step 3 by removing D and E which are dominated by AB (i.e. $area(D) = area(E) = 3 < area(AB) = 6$). Finally, the last step computes the indistinct patterns of skypatterns. Note that this step in this example is unnecessary because the area is strictly $\{freq\}$-skylineable.

6. Mining skypatterns using dynamic CSP

This section describes how the skypattern mining problem can be modeled and solved by using DynCSP [52, 53]. A major advantage of this method is that it improves the mining step during the process thanks to constraints dynamically arising from the current set of candidate skypatterns. These constraints avoid producing new solutions dominated by the current skypatterns and thus reduce the search space. More precisely, each time a solution is found, a new constraint is dynamically posted. The process stops when we cannot enlarge the dominated area further (cf. Figure 1). The set of obtained representative patterns is a subset of the set of representative patterns extracted with Aetheris (cf. Section 3.3). The completeness of our CP+Sky method is insured by the completeness of the CSP solver. The implementation has been carried out in `Gecode`[3].

The rest of this section is organized as follows. Subsection 6.1 recalls some general background on CSP and DynCSP. Subsection 6.2 describes how skypat-

[3] `http://www.gecode.org/`

tern mining can be modeled using DynCSP. Subsection 6.3 presents the pattern encoding as well as the filtering that is achieved. Subsections 6.4 and 6.5 are devoted to closedness constraints and freeness constraints. Finally, Subsection 6.6 provides an example.

6.1. CSP and DynCSP

Constraint Satisfaction Problem. A CSP [54, 55] $P{=}(\mathcal{X},\mathcal{D},\mathcal{C})$ is defined by a finite set of variables $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$, a set of domains $\mathcal{D}$ which maps every variable $x_i \in \mathcal{X}$ to a finite set of values $D(x_i)$ and a finite set of constraints $\mathcal{C}$.

Algorithm 1 provides a general overview of a CSP solver. Dom and $Store$ denote respectively the current domains and the current set of constraints. Essentially, a CSP solver consists of a depth-first search algorithm. At each node of the search tree, procedure $Constraint$-$Search$ selects an unassigned variable (line 5) according to user-defined heuristics[4] and assigns it a value (line 6). After that assignment, procedure $Constraint$-$Search$ is called recursively (line 7). It backtracks when a constraint cannot be satisfied, i.e. when at least one domain is empty (line 2). A solution is obtained (line 9) when each domain $Dom(x_i)$ is reduced to a singleton and all constraints are satisfied.

```
Algorithm 1: Constraint-Search($Dom$)
1  $Dom \leftarrow Filtering(Dom, Store)$;
2  if there exists $x_i \in \mathcal{X}$ s.t. $Dom(x_i)$ is empty then
3      return failure;
4  if there exists $x_i \in \mathcal{X}$ s.t. $|Dom(x_i)| > 1$ then
5      Select $x_i \in \mathcal{X}$ s.t. $|Dom(x_i)| > 1$;
6      forall the $v \in Dom(x_i)$ do
7          $Constraint$-$Search$($Dom \cup \{x_i \rightarrow \{v\}\}$);
8  else
9      output solution $Dom$;
10     $Store \leftarrow Store \cup \{\phi(\mathcal{X})\}$;
```

The main concept used to speed-up the search is the constraint propagation by *Filtering* method. This method reduces the domains of variables such that

[4] For our implementation, we used the most constrained variable order heuristics, which branches over the variable contained in most constraints; this order is dynamic (updated during the search).

they remain locally consistent. Constraint propagation operates on an individual constraint. To maintain local consistency for individual constraints, propagation rules are used. Given a constraint and the current domains of the variables in its scope, a propagator removes domain values that do not satisfy the constraint. Since variables usually participate in several constraints, the updated domains are propagated to the other constraints, whose propagators are in turn activated. This process of constraint propagation is repeated for all constraints until no more domain values can be removed or a domain becomes empty.

Dynamic CSP. A DynCSP [52, 53] is a sequence $P_1, P_2, ..., P_n$ of CSPs, each one resulting from some changes in the definition of the previous one. These changes may affect every component in the problem definition: variables (additions or removals), domains (value additions or removals), constraints (additions or removals).

6.2. DynCSP-based method for mining skypatterns

This section provides our CSP-based method CP+Sky for mining skypatterns. As before, the representative patterns are searched according to M' using the skylineability principle. The key idea is to use constraints on the dominance relation, which are dynamically added during the mining process. These constraints avoid producing solutions dominated by the solutions already extracted and thus reduce the search space. We start by highlighting the way we handle DynCSP and then we provide our encoding.

For our approach, changes between CSP P_i and CSP P_{i+1} are only performed by adding new constraints without any removal of constraints. Additions are handled in a straightforward way with the help of filtering. Solving such a DynCSP involves solving a single CSP with additional constraints posted during search. These constraints will survive backtracking and state that next solutions should verify both the current set of constraints as well as the added ones. Each time a new solution is found, new constraints $\phi(\mathcal{X})$ are imposed. Such constraints state that next solutions should verify both the current set of constraints $Store$ and $\phi(\mathcal{X})$ (cf. line 10 of Algorithm 1).

Variable *x* will denote the (unknown) skypattern we are looking for. We consider the sequence $P_1, P_2, \ldots, P_n$ of CSP where each $P_i = (\{x\}, \mathcal{L}, q_i(x))$ and:

$$\begin{array}{rcll} q_1(x) & = & dis_\theta(x) & \text{where } dis_\theta(x) \text{ denotes the representative pattern } x. \\ q_{i+1}(x) & = & q_i(x) \wedge \phi_i(x) & \text{where } s_i \text{ is the first solution to query } q_i(x). \end{array}$$

First, the constraint $dis_\theta(x)$ states that *x* must be either a closed pattern w.r.t. M' (i.e. $dis_\theta(x) = closed_{M'}(x)$) or a free pattern w.r.t. M' (i.e. $dis_\theta(x) = free_{M'}(x)$).

Then, the constraint $\phi_i(x) \equiv (s_i \not\succ_M x)$ states that the next solution (which is searched for) will not be dominated by s_i. Using a short induction proof, we can easily argue that query $q_{i+1}(x)$ looks for a pattern *x* that will not be dominated by any of the patterns s_1, s_2, . . ., s_i.

Each time the first solution s_i to query $q_i(x)$ is output by Algorithm 1, we dynamically post a new constraint $\phi_i(x)$ (see line 10) leading to a reduction of the search space. For skypatterns, $\phi_i(x)$ states that $(s_i \not\succ_M x)$:

$$\phi_i(x) \equiv \left(\bigvee_{m \in M} m(s_i) < m(x) \right) \vee \left(\bigwedge_{m \in M} m(s_i) = m(x) \right)$$

This process stops when we cannot enlarge the dominated area further, i.e. there exists n s.t. query $q_{n+1}(x)$ has no solution. The dominated area cannot be extended and is fully established.

But, the n extracted patterns s_1, s_2, . . ., s_n are not necessarily all skypatterns. Some of them could only be "intermediate" patterns simply used to enlarge the dominated area. A post processing step must be performed to filter all candidate patterns s_i that are not skypatterns, i.e. for which there exists s_j $(1 \leq i < j \leq n)$ s.t. s_j dominates s_i. So mining skypatterns is achieved in a two-step approach:

1. Compute the set $S = \{s_1, s_2, \ldots, s_n\}$ of candidates using DynCSP.
2. Filter all patterns $s_i \in S$ that are not skypatterns.

While the number of candidates (n) could be very large, it remains reasonably-sized in practice for the experiments we conducted (see Section 7).

However, the order in which candidates are produced in the first step influences the way the dominated area is enlarged, and therefore the effectiveness of CP+Sky. One way to enhance the efficiency would be to select the most beneficial order in which candidates are generated. In the case of a single measure m, we can always derive an optimal order that guarantees that any solution produced in step 1 is a skypattern, thus avoiding the need for post-processing. It suffices to generate the patterns from the largest values of m to the lowest values of m (cf. Definition 1). But in the general case where several measures are involved, finding such an optimal order is often impossible and post-processing is required.

6.3. Pattern encoding and filtering

We now introduce the modeling of a pattern that can be provided to the constraint programming system. Let ***d*** be the 0/1 matrix where, for each transaction

t and each item i, $(d_{t,i} = 1)$ iff $(i \in t)$. Pattern variables are set variables represented by their characteristic function with Boolean variables. [35] and [36] model an unknown pattern *x* and its associated dataset $\mathcal{T}$ by introducing two sets of Boolean variables:

- item variables $\{X_i \mid i \in \mathcal{I}\}$ where $(X_i = 1)$ iff $(i \in x)$,
- transaction variables $\{T_t \mid t \in \mathcal{T}\}$ where $(T_t = 1)$ iff $(x \subseteq t)$.

Each set of Boolean variables aims to represent the characteristic function of the unknown pattern.

The relationship between *x* and $\mathcal{T}$ is modeled by posting reified constraints stating that, for each transaction t, $(T_t = 1)$ iff *x* is a subset of t:

$$\forall t \in \mathcal{T}, (T_t = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} X_i \times (1 - d_{t,i}) = 0 \quad (1)$$

A reified constraint associates a 0/1 variable to a constraint reflecting whether the constraint is satisfied (value 1) or not (value 0). Such constraints are useful for expressing propositional formulas over constraints and for expressing that a certain number of constraints hold. Reified constraints do not enjoy the same level of propagation as simple constraints, but if the solver deduces $T_t = 1$ (resp. $T_t = 0$), then the sum must be equal to 0 (resp. must be different from 0).

The propagation is also performed in the same way from the sum constraint towards the equality constraint. For example, when an item variable X_i is set, the following propagation is applied to the reified constraints described by equation 1 (see [35] for more details):

- if for some t, $\sum_{i \in \mathcal{I}}(\min D(X_i)) \times (1 - d_{t,i}) > 0$ then remove 1 from $D(T_t)$,
- if for some t, $\sum_{i \in \mathcal{I}}(\min D(X_i)) \times (1 - d_{t,i}) = 0$ then remove 0 from $D(T_t)$.

Finally, using the Boolean encoding, it is worth noting that some measures are easy to encode: $\mathit{freq}(x) = \sum_{t \in \mathcal{T}} T_t$ and $length(x) = \sum_{i \in \mathcal{I}} X_i$. So, the minimal frequency constraint $\mathit{freq}(x) \geq \theta$ (where θ is a threshold) is encoded by the constraint $\sum_{t \in \mathcal{T}} T_t \geq \theta$. In the same way, the maximal size constraint $length(x) \leq \alpha$ (where α is a threshold) is encoded by the constraint $\sum_{i \in \mathcal{I}} X_i \leq \alpha$.

6.4. Closedness constraints

This section describes how to encode $closed_{M'}(x)$. As an illustration, we provide the examples of M'=$\{min\}$ and M'=$\{freq\}$. Recalling that thanks to skylineability, these measures also allow to handle measures such as $mean$, $area$, $growth$-$rate$, etc. In practice, these two examples are enough for running the experiments given in Section 7.

Let M'=$\{min\}$ and $val(j)$ a function that associates an attribute value to each item j. If item i belongs to *x*, then its value must be greater than or equal to the min. Conversely, if this value is greater than or equal to the min, then i must belong to *x* (if not, *x* would not be maximal for inclusion). Item i belongs to *x* is encoded as $(X_i = 1)$. So, *x* is a closed pattern for the measure min iff:

$$\forall i \in \mathcal{I}, (X_i = 1) \Leftrightarrow val(i) \geq min\{val(j) \mid j \in x\} \tag{2}$$

Let M'=$\{freq\}$, the closedness constraint ensures that a pattern has no superset with the same frequency. If item i belongs to *x*, it is obvious that $freq(x \cup \{i\}) = freq(x)$. Conversely, if $freq(x \cup \{i\}) = freq(x)$, then i must belong to *x* (if not, *x* would not be maximal for inclusion). $freq(x)$ is encoded as $\sum_{t\in\mathcal{T}} T_t$ and $freq(x \cup \{i\})$ is encoded as $\sum_{t\in\mathcal{T}} T_t \times d_{t,i}$. Finally, the constraint $closed_{M'}(x)$ is encoded using Equation (1) and Equation (3).

$$\forall i \in \mathcal{I}, (X_i = 1) \Leftrightarrow \sum_{t\in\mathcal{T}} T_t \times (1 - d_{t,i}) = 0 \tag{3}$$

6.5. Freeness constraints

Similar to the closedness constraint, we now give two examples of encoding of the freeness constraint. Following our principle of dealing with measures used in the experiments, we give the examples of M'=$\{freq\}$ and M'=$\{max\}$ because applying the minimal converter $\underline{c}$ on $mean$ gives max (cf. Table 4).

Let M'=$\{max\}$ and $val(j)$ a function that associates an attribute value to each item j. If item i does not belong to *x*, then its value must be greater than or equal to the min. Conversely, if this value is greater than or equal to the min, then i cannot belong to *x* (if i belongs to *x*, *x* would not be minimal for inclusion). Item i does not belong to *x* is encoded as $(X_i = 0)$. So, *x* is a free pattern for the measure max iff:

$$\forall i \in \mathcal{I}, (X_i = 0) \Leftrightarrow val(i) \leq max\{val(j) \mid j \in x\} \tag{4}$$

Let M'=$\{freq\}$, the freeness constraint ensures that a pattern has no subset with the same frequency. If item i does not belong to *x*, it is obvious that $freq(x \setminus$

$\{i\}) = freq(x)$. Conversely, if $freq(x \setminus \{i\}) = freq(x)$, then i must not belong to *x* (if i belongs to *x*, *x* would not be minimal for inclusion).

In order to encode $freq(x \setminus \{i\})$, we introduce Boolean variables $T'_{t,i}$ such that the relationship between $x \setminus \{i\}$ and $\mathcal{T}$ is modeled by posting reified constraints stating that, for each transaction t, $(T'_{t,i} = 1)$ iff $x \setminus \{i\}$ is a subset of t:

$$\forall t \in \mathcal{T}, (T'_{t,i} = 1) \Leftrightarrow \sum_{j \in \mathcal{I} \setminus \{i\}} X_j \times (1 - d_{t,j}) = 0 \tag{5}$$

$freq(x)$ is encoded as $\sum_{t \in \mathcal{T}} T_t$ and $freq(x \setminus \{i\})$ is encoded as $\sum_{t \in \mathcal{T}} T'_{t,i}$. So $free_{M'}(x)$ is modeled using Equation (1) and Equation (6).

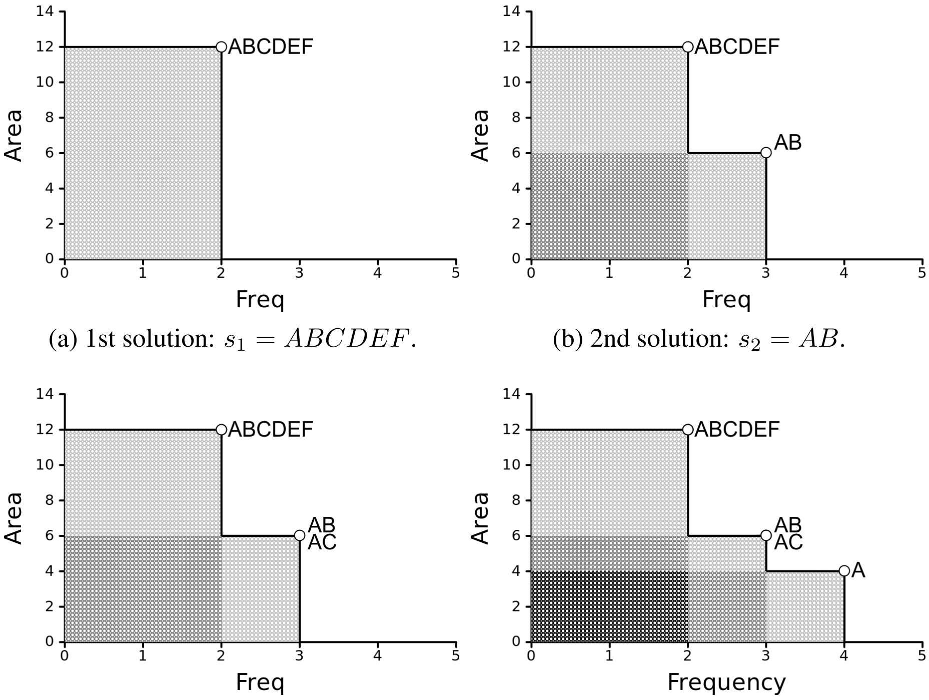
$$\forall i \in \mathcal{I}, (X_i = 0) \Leftrightarrow \sum_{t \in \mathcal{T}} (T_t - T'_{t,i}) = 0 \tag{6}$$

6.6. Solving the running example using DynCSP

We now illustrate CP+Sky on the running example in Table 1a with M=$\{$*freq*$, area\}$. We use the maximal converter $\overline{c}$ and thus the closedness constraint. As $\overline{c}(area) = freq$, we get M'=$\{$*freq*$\}$. Figure 5 depicts the various steps of the resolution.

Let P_1 be the associated DynCSP (see Section 6.2). $P_1 = (\{x\}, \mathcal{L}, q_1(x))$ where query $q_1(x) = closed_{M'}(x)$. Its first solution is pattern $s_1 = ABCDEF$ (with *freq*$(s_1) = 2$ and $area(s_1) = 12$), cf. Figure 5a. So, we consider query $q_2(x) = closed_{M'}(x) \wedge (s_1 \not\succ_M x)$ stating that we are looking for a closed pattern *x* not dominated by $s_1 = ABCDEF$. Its first solution is pattern $s_2 = AB$ (with *freq*$(s_2) = 3$ and $area(s_2) = 6$), cf. Figure 5b. Then, the next query is $q_3(x) = closed_{M'}(x) \wedge (s_1 \not\succ_M x) \wedge (s_2 \not\succ_M x)$ stating that we are looking for a closed pattern *x* neither dominated by s_1 nor s_2. Its first solution is pattern $s_3 = AC$ (with *freq*$(s_3) = 3$ and $area(s_3) = 6$), cf. Figure 5c. The next query is $q_4(x) = q_3(x) \wedge (s_3 \not\succ_M x)$ whose first solution is $s_4 = A$ (cf. Figure 5d) and then query $q_5(x) = q_4(x) \wedge (s_4 \not\succ_M x)$. $q_5(x)$ has no solution since the dominated area cannot be enlarged further and the process ends at $n = 5$.

In this example, note that all extracted patterns are skypatterns (i.e. there are no intermediate patterns). The CSP system did not generate solutions that do not satisfy the dominance relation. Experiments in the next section provide examples with intermediate patterns.



(a) 1st solution: $s_1 = ABCDEF$.

(b) 2nd solution: $s_2 = AB$.

(c) 3rd solution: $s_3 = AC$.

(d) Last solution: $s_4 = A$.

Figure 5: Solving the running example using DynCSP.

7. Experiments

We first report an experimental study on several UCI benchmarks (see Section 7.1). We then discuss the practical use of skypatterns in a chemoinformatics case study (see Section 7.2). All experiments were conducted on a personal computer running a Linux operating system with an i3 core processor with a clock speed of 2.13 GHz and 4GB of RAM. The implementation of Aetheris refers to [1]. The implementation of CP+Sky was carried out in `Gecode`. All source codes and data sets are publicly available at `https://forge.greyc.fr/projects/skymining/files`

Note that it was shown in [1] that Aetheris always outperforms a baseline approach by at least a factor of 10. In addition, the collection of skypatterns is always much smaller than the collection of patterns returned by an optimal constraint-ased approach (i.e. assuming that an ideal end-user is able to perfectly set the thresholds and then run a usual constraint-based mining method). Therefore, this

Dataset				$M_6 = \{freq, max, area, mean, growth\ rate\}$					Averages over $\{M_1, M_2, M_3, M_4, M_5\}$				
					CP+Sky		Aetheris			CP+Sky		Aetheris	
	# items	# transactions	density	# of Skypatterns	# of Candidates	Time (sec)	# of Closed Patterns	Time (sec)	# of Skypatterns	# of Candidates	Time (sec)	# of Closed Patterns	Time (sec)
abalone	28	4,177	0.321	76	5,255	25	9,947	**1**	43.20	3,393.20	6	9,808.80	**1**
anneal	68	798	0.195	187	13,903	12	35,152	**3**	80.60	6,748.20	6	30,606.00	**1**
austral	55	690	0.272	172	49,379	24	243,156	**20**	77.00	19,626.80	18	228,115.20	**13**
breast	43	286	0.231	38	2,311	**1**	7,721	**1**	23.60	1,374.20	**1**	7,369.20	**1**
cleve	43	303	0.325	97	19,370	**8**	77,203	**8**	51.80	11,215.80	5	74,670.80	**4**
cmc	28	1,473	0.357	62	12,760	12	25,649	**1**	36.00	7,702.80	9	25,408.60	**1**
crx	59	690	0.269	143	78,327	**44**	349,721	55	59.40	31,482.80	**19**	317,090.20	27
german	76	1,000	0.276	308	347,957	**426**	3,662,911	652	121.80	148,255.40	**249**	3,525,072.40	305
glass	34	216	0.295	52	2,633	**1**	7,165	**1**	31.80	1,587.80	**1**	6,920.80	**1**
heart	38	270	0.368	154	16,960	**6**	72,618	8	73.60	9,118.00	**4**	70,124.20	**4**
hepatic	45	155	0.421	237	41,096	**10**	222,333	31	103.20	16,156.20	**4**	206,742.00	13
horse	75	300	0.235	63	28,275	**27**	191,177	47	34.20	12,609.60	**16**	182,982.20	23
hypo	47	3,163	0.389	478	221,032	**469**	1,604,864	893	204.80	145,359.40	**303**	1,565,797.20	481
iris	15	150	0.333	7	86	**1**	93	**1**	5.80	72.40	**1**	91.80	**1**
lymph	59	142	0.322	161	26,200	**7**	116,030	26	64.00	12,532.40	**4**	105,260.00	11
mushroom	119	8,124	0.193	176	14,599	1,186	1,153,229	**548**	75.40	5,767.60	1137	995,808.40	**221**
new-thyroid	21	215	0.287	15	200	**1**	288	**1**	12.00	151.80	**1**	285.00	**1**
page	35	941	0.314	92	4,482	8	21,121	**2**	49.00	2,576.00	6	20,207.80	**1**
pima	26	768	0.346	53	1,400	3	12,559	**1**	32.40	943.80	3	12,439.40	**1**
tic-tac-toe	29	958	0.344	82	11,206	11	43,318	**3**	51.20	7,867.40	9	4 2,902.20	**2**
vehicle	58	846	0.327	280	164,152	172	745,353	**138**	126.20	73,390.60	**69**	689,937.80	84
wine	45	178	0.311	43	7,780	**2**	36,671	4	25.40	4,538.60	**2**	34,397.00	**2**
zoo	43	101	0.394	56	4,654	**1**	14,431	**1**	34.20	2,642.60	**1**	12,851.20	**1**

Table 5: Comparing the two methods on 23 UCI datasets (a detailed summary).

empirical study focuses on the comparison between Aetheris and CP+Sky.

7.1. Experiments on UCI benchmarks

7.1.1. Experimental protocol

We focused on 23 different (in terms of dimensions and density[5]) datasets (see the left column in Table 5) from the UCI[6] repository. We considered the set of measures M=$\{freq$, max, $area$, $mean$, $growth$-$rate\}$ and selected 6 subsets: M_1=$\{freq$,$area$, $mean$, $growth$-$rate\}$, M_2=$\{freq$, max, $area$, $growth$-$rate\}$, M_3=$\{freq$, max, $area$, $mean\}$, M_4=$\{freq$, max, $mean$, $growth$-$rate\}$, M_5=$\{max$, $area$, $mean$, $growth$-$rate\}$ and M_6=M. Measures using numeric values, like $mean$ or max, were applied to randomly generated attribute values (within the range $[0, 1]$).

For each method, we report the CPU-time and the number of skypatterns for every query on the selected set of measures. Note that Aetheris first computes the set of closed patterns[7] w.r.t. M' and then applies the $\mathcal{S}ky$ operator on the extracted collection. On the other hand, CP+Sky does not mine closed patterns as a first step but instead computes a small set of candidates using DynCSP and then applies the $\mathcal{S}ky$ operator. For each method, the reported CPU-times include the different processing steps. We also report for each dataset the size of the condensed representation w.r.t M' and the number of candidates to respectively analyse the behaviors of Aetheris and CP+Sky.

7.1.2. Performance analysis

A general overview. Table 5 provides the results of CPU-times for CP+Sky and Aetheris for 138 skypattern queries (23 × 6). We report for each dataset and for every collection of measures[8]:

- the number of skypatterns (the largest output is 478),
- with CP+Sky: the number of candidates and the associated CPU-time,
- with Aetheris: the number of closed patterns and the associated CPU-time.

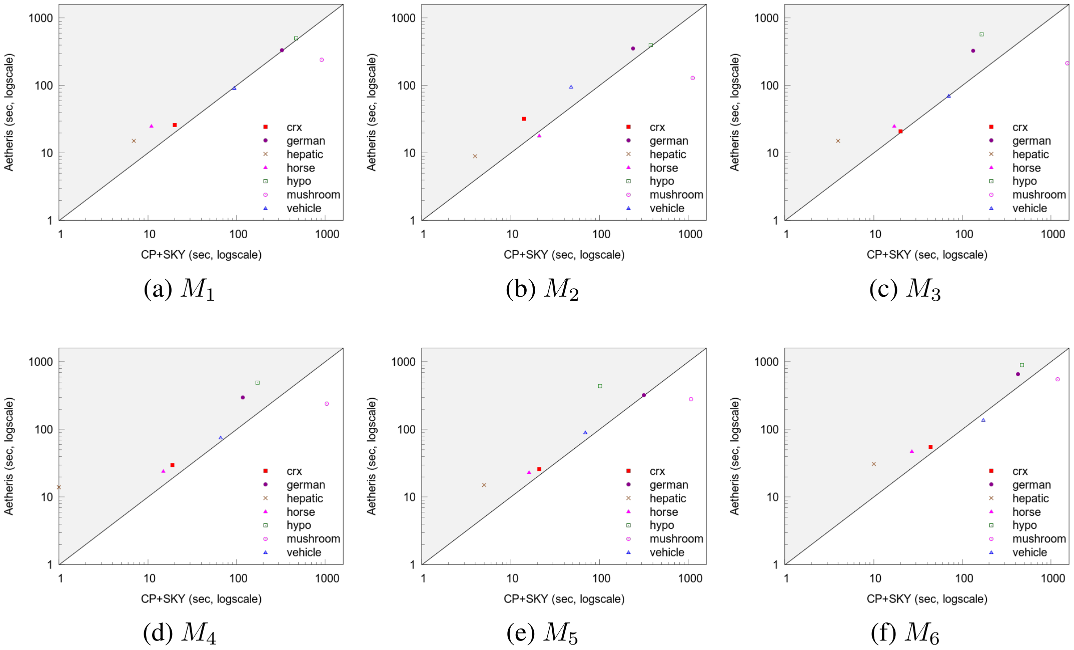
Table 5 indicates that CP+Sky and Aetheris run within the same order of magnitude. On 16 datasets out of 23, CPU times for both CP+Sky and Aetheris are very small (less than 30 seconds). The results of the 7 remaining data sets are analyzed in more detail to highlight the differences and limitations of our proposed methods.

[5] The density of a dataset is $\sum_{t\in\mathcal{T}}|t|/(|\mathcal{T}| \times |\mathcal{I}|)$.

[6] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

[7] We use an absolute minimal frequency threshold of 1.

[8] Reported values in columns (6-9) are associated to M_6. But, reported values in columns (10-14) represent average values over M_1, M_2, M_3, M_4 and M_5.



(a) M_1 (b) M_2 (c) M_3

(d) M_4 (e) M_5 (f) M_6

Figure 6: Comparing CPU times on the 7 selected datasets for $M_i, i \in [1..6]$.

A more detailed analysis. Figure 6 depicts a scatter plot of CPU-times for CP+Sky and Aetheris. Each point represents a skypattern query for one of the 7 selected datasets: its x-value (log-scale) is the CPU-time obtained with CP+Sky, its y-value (log scale) the CPU-time with Aetheris. A specific color is associated to each dataset. The line $y = x$ draws the case where Aetheris and CP+Sky have the same CPU-times. Most of the points are above this line, which means a longer runtime for Aetheris. With all the measures (i.e. M_6), the speed-up is 1.9 (resp. 1.53) on `hypo` (resp. `german`). The only exception is the `mushroom` dataset.

Another interesting result provided by Table 5 is the number of closed patterns extracted by Aetheris in comparison with the number of candidates generated by CP+Sky. The number of candidates remains small (in the thousands) compared to the number of closed patterns (in millions). Figure 7 illustrates this particular result for the selected datasets. It reports for each set of measures M_i ($i \in [1, 6]$) and the 7 datasets investigated in Figure 6, the number of closed patterns, the number of candidate patterns and the number of skypatterns (both methods). This figure highlights the discrepancy between the methods with the distinct lower number of representative patterns required by CP+Sky in comparison to the Aetheris method.

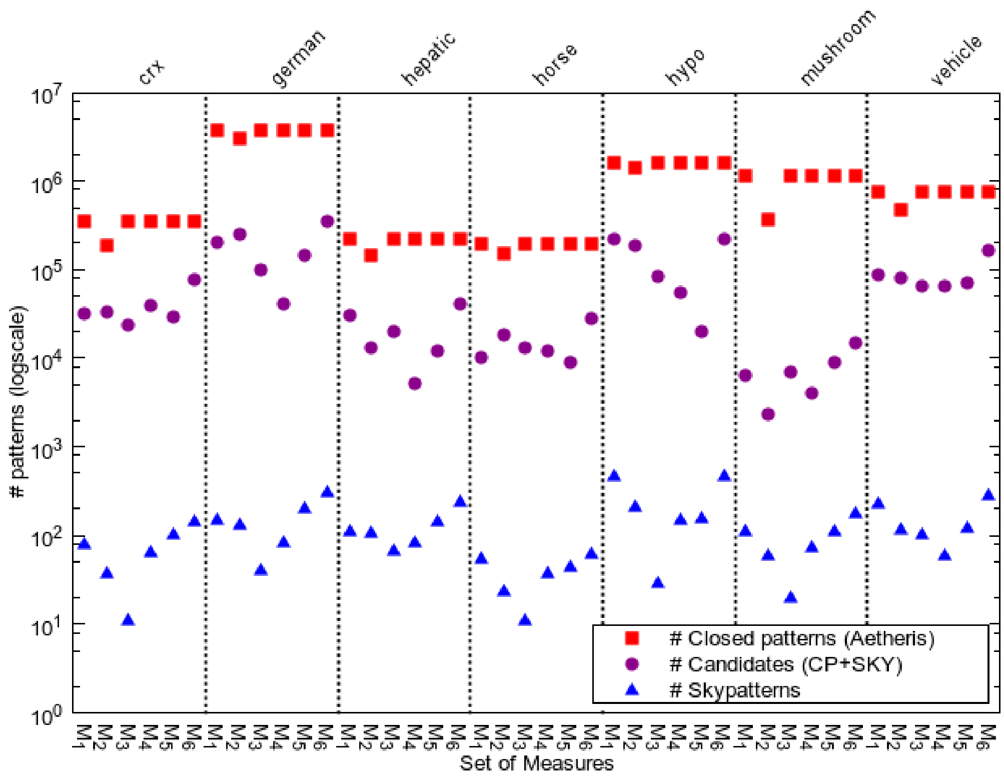


Figure 7: Comparing # of patterns on the 7 selected datasets for $M_1, \dots, M_6$.

Dataset	# of Sky-patterns	# of Candidates Patterns	CPU-Times (Seconds)			# of Closed Patterns	CPU-Times (Seconds)		
			Step 1	Step 2	Total		Step 1	Step 2	Total
crx	143	78,327	44	0	**44**	349,721	40	15	55
german	308	347,957	425	1	**426**	3,662,911	511	141	652
hepatic	237	41,096	10	0	**10**	722,333	23	8	31
horse	63	28,275	26	1	**27**	191,177	33	14	47
hypo	478	221,032	468	1	**469**	1,604,864	812	83	893
mushroom	176	14,599	1,186	0	1186	1,153,229	497	51	**548**
vehicle	280	164,152	171	1	**172**	745,353	111	27	138

Table 6: Comparing the CPU times for the two steps on the 7 selected datasets for M_6.

Table 6 shows an in-depth comparison of the CPU-times for the two steps performed by CP+Sky and Aetheris for the set of measures M_6. The second (i.e. the post-processing) step is the same for both methods and is performed using the same classical algorithm: the BNL approach [21]. The time complexity of this approach is $\mathcal{O}(n^2)$ where n is the number of representative patterns generated in the first step. These results clearly show that CP+Sky takes less time than Aetheris to generate the representative patterns. This is in part explained by the huge number of closed patterns that Aetheris needs to post-process. This drawback does not exist for CP+Sky because the number of candidates remains small and thus, the post-processing step is negligible. The only exception is for the `mushroom` dataset, where Aetheris is very efficient

7.1.3. Summary

No method is constantly better on all datasets. CP+Sky usually generates a low number of candidates compared to Aetheris. The number of candidates and closed patterns seems to provide an appropriate explanation of the performances of the two methods. However, they only constitute simple indicators and do not take into account other evidence. For instance the results on the `mushroom` dataset may seem surprising and counter-intuitive. Even if the number of candidates $(14,599)$ is low compared to the number of closed patterns extracted from this dataset $(1,153,229)$, Aetheris is more performant than CP+Sky.

The `mushroom` dataset (which is the largest UCI dataset both in terms of transactions and items) has the lowest density (around 18%), which implies that its number of closed patterns is small w.r.t its size. The same reasoning also applies for the number of candidates. In this case, the size of the set of constraints is important, as there are as many reified constraints as transactions (cf. Section 6.3). So, filtering takes much more time to generate the candidates. For this dataset, it is more efficient to compute the closed patterns and filter them, even if they are more numerous.

Following the `mushroom` dataset analysis, we investigated the notion of density and its impact on the performances of our two methods by generating several datasets and varying their density from 0.15 to 0.65 (we kept the numbers of items and transactions similar to the `mushroom` dataset). Figure 8 shows the evolution of CPU-times according to the density for both methods. As the items in the data are randomly generated to provide the density value, the number of closed patterns increases according to the density. Thus, the running time of Aetheris also increases according to this parameter. The behavior of CP+Sky is more complex due to the dynamic pruning. Having more candidate skypatterns is a benefit if these patterns are able to prune significantly the search space. The experimental study shows a good trade-off with a density close to 0.5 (note that in practice there are very few real-world datasets with a density higher than 0.5).

We also performed experiments on other data sets from the UCI repository, such as the chess (75 items, 3196 transactions, density 0.49) and the connect data sets (129 items, 67557 transactions, density 0.33). However, we were not able

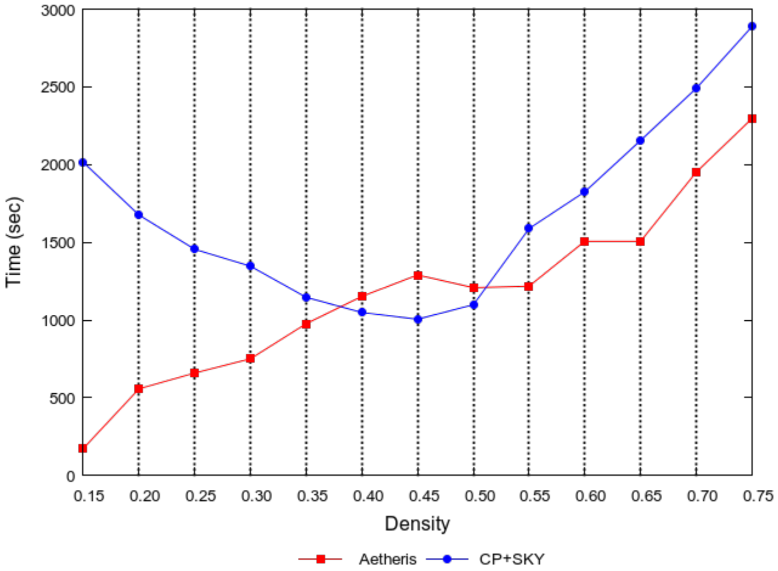


Figure 8: Measuring the impact of the density for the two methods.

to complete the skypattern mining process. Aetheris was not able to complete the closed pattern mining step on either data set. For CP+Sky, the first step of candidates generation took approximatively 20 hours and generated more than 40 million candidates for the chess dataset. The application of the sky operator failed because of the quadratic complexity (i.e. $\mathcal{O}(n^2)$) of the BNL method. The experiments on the connect dataset with CP+Sky were aborted after more than 24 hours of computation.

7.2. Case study: discovering toxicophores

A major issue in chemoinformatics is to establish relationships between chemicals and their activity in (eco)toxicity. Chemical fragments[9] which cause toxicity are called *toxicophores* and their discovery is at the core of prediction models in (eco)toxicity [56]. The aim of this study, which is part of a larger research collaboration with the CERMN Lab, is to investigate the use of skypatterns for discovering toxicophores.

[9] A fragment denotes a connected part of a chemical structure having at least one chemical bond.

7.2.1. *Experimental protocol*

The dataset was collected from the ECB web site[10]. For each chemical, the chemists have associated it with hazard statement codes (HSC) in 3 categories: H400 (very toxic, CL50 $\leq$ 1 mg/L), H401 (toxic, 1 mg/L $<$ CL50 $\leq$ 10 mg/L), and H402 (harmful, 10 mg/L $<$ CL50 $\leq$ 100 mg/L). We focus on the H400 and H402 classes. The dataset $\mathcal{T}$ consists of 567 chemicals (transactions), 372 from the H400 class and 195 from the H402 class. The chemicals are encoded using 1450 frequent closed subgraphs (items) previously extracted[11] with a 1% relative frequency threshold. Therefore, the extracted skypatterns correspond to sets of chemical fragments, which are represented by frequent closed subgraphs [57].

Discovering candidate toxicophores is similar to supervised descriptive rule discovery [11], and or learning classification rules, and we therefore use *growth rate* as a contrast measure. Indeed, when a pattern's frequency strongly increases from class H402 to class H400, it can be considered a potential structural alert related to toxicity. If a compound includes several such fragments in its graph structure, it is more likely to be toxic. Emerging patterns model this idea using the growth rate measure. On the other hand, real-world datasets are often noisy and patterns with low frequency may be artefacts. We also use the *frequency* measure to ensure the representativeness of the patterns (i.e. the higher the frequency, the better).

The skypattern framework makes it possible to integrate measures coming from the background domain. In ecotoxicity, chemists know that the *aromaticity* measure is a chemical property that favors toxicity since their metabolites can lead to very reactive species which can interact with biomacromolecules in a harmful way (the higher the aromaticity, the higher the toxicity hypothesis). The chemical knowledge provides the aromaticity of the chemical fragments and we compute the aromaticity of pattern as the mean of the aromaticity of its chemical fragments.

We tested several combinations of measures: M_1=$\{growth\text{-}rate, freq\}$, M_2=$\{growth\text{-}rate, \quad aromaticity\}$, M_3=$\{freq, aromaticity\}$ and M_4=$\{growth\text{-}rate, freq, aromaticity\}$.

7.2.2. *Performance analysis*

Table 7 reports, for each set of measures $M_i \in [1..4]$: (i) the number of skypatterns, (ii) for CP+Sky, the number of candidates and the associated CPU-time

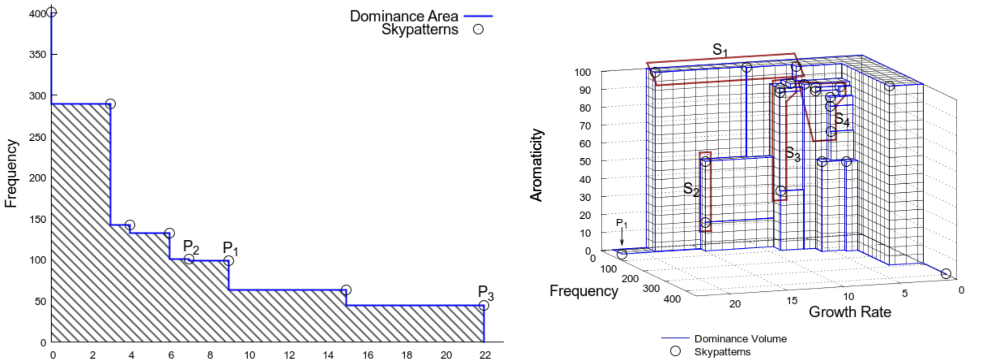
[10] European Chemicals Bureau: `http://echa.europa.eu/`

[11] A chemical Ch contains an item A if Ch supports A, and A is a frequent subgraph of $\mathcal{T}$.

	Skypatterns				
	# of Sky-patterns	CP+Sky		Aetheris	
		# of Candidates	CPU-Time	# of closed patterns	CPU-Time
M_1={*growth-rate*, *freq*}	**8**	613	**18m:34s**	41, 887	**19m:20s**
M_2={*growth-rate*, *aromaticity*}	**5**	140	**15m:32s**	53, 201	**21m:33s**
M_3={*freq*, *aromaticity*}	**2**	456	**16m:45s**	157, 911	**21m:16s**
M_4={*growth-rate*, *freq*, *aromaticity*}	**21**	869	**17m:49s**	69, 827	**21m:40s**

Table 7: Skypattern mining on ECB dataset.

and (iii) for Aetheris, the number of closed patterns and the associated CPU-time. CP+Sky outperforms Aetheris in term of CPU-times. Moreover, the number of candidates for CP+Sky is drastically smaller than the number of closed patterns computed by Aetheris. It clearly shows the usefulness of filtering via the dynamically posted constraints.



(a) Distribution of the skypatterns for M_1.

(b) Distribution of the skypatterns for M_4.

Figure 9: Analysing the skypatterns.

7.2.3. *Qualitative analysis*

We now analyse the skypatterns qualitatively by comparing them with well-known environmental toxicophores [58]. With the growth rate and frequency measures (i.e. M_1), only 8 skypatterns are found, among those we have emphasized 3 well-known toxicophores. Figure 9a depicts these skypatterns denoted

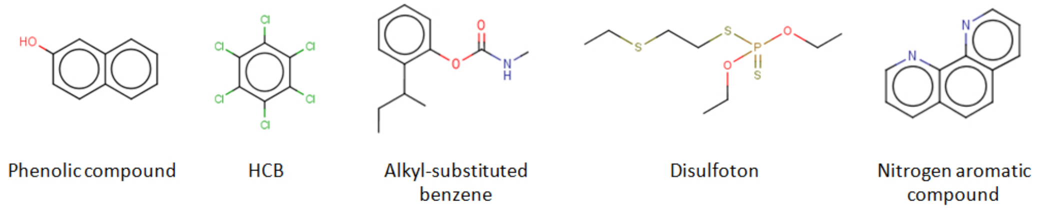


Figure 10: Examples of environmentally hazardous compounds related to skypatterns S4, S3, S2, P1, and S1.

P_i, one of them is on the y-axis. Two of them are components of widespread pesticides, namely the chloro-substituted aromatic ring (P_1: {`Clc`}) and organophosphorus moiety (P_3: {`OP, OP=S`}). The third one, the phenol ring (P_2: {`c1(ccccc1)O`}) is related to hydrophobocity and formation of free radicals [59].

The most interesting results follow from the addition of the background knowledge. Indeed, adding the aromaticity measure leads to skypatterns with novel chemical characteristics. We discuss the results obtained with the growth rate, frequency and aromaticity measures (i.e. M_4). Once again, the whole set of skypatterns remains small and can lends itself to straight-forward analysis. 21 skypatterns are mined (see Figure 9b). To simplify the picture, the S_i denote sets of skypatterns sharing a common chemical feature. The figure emphasizes several environmentally hazardous chemical fragments: the phenol ring (S_4), the chloro-substituted aromatic ring (S_3), the alkyl-substituted benzene (S_2), and the organophosphorus moiety (P_1). Besides, information dealing with nitrogen aromatic compounds is also extracted (S_1). The comparison of this list of patterns with jumping emerging fragments (JEF) extracted from previous experiments [60] highlights the generalization potency of the skypatterns. As an example, the organophosphorus moiety skypattern is a generalization of around 90 JEFs and can be seen as a kind of maximum common structure (i.e. consensus structure) of these fragments.

The main result of the study concerns the chemical interpretation of the outputs. Indeed, the generalization capability of the skypatterns leads to a reduced list of potential toxicophores easily interpretable by the chemists (cf. Figure 10). Even if the skypattern process is complete with respect to the user preferences and the dataset, the proposed list of toxicophores depends on the composition of the used dataset and the measures. We cannot hope to discover all the ecotoxicological structural alerts in one pass, because we cannot expect that all the structural alerts are present in a single dataset. But any progress in the discovery of potential toxicophores is a valuable step. The method can suggest new compounds as toxicophores, i.e. toxicophores which were previously unknown. Further in vitro experiments are required to validate such candidate toxicophores.

Note that adding the p-value as a measure in order to provide a statistical significance in the skypatterns does not change the results in this experiment. Indeed, p-value is maximized by the closed patterns adequate to frequency and with the set of measures $M_5 = \{growth\text{-}rate, freq, aromaticity, p\text{-}value\}$ only the aromaticity leads to another closed patterns. Out of curiosity, we ran the experiment with M_5 and we get 28 skypatterns instead of 21 with M_4 (the 21 skypatterns extracted with M_4 are still skypatterns with M_5). There are no significant new insights from a chemical point of view.

8. Conclusion and Perspectives

In this paper, we investigate in detail the skyline pattern mining problem by studying the theoretical relationships between condensed representations of patterns and skypatterns. Based on the concept of skylineability, we have devised the static method Aetheris and the dynamic method CP+Sky. Aetheris exploits the condensed representations of patterns to provide a proper superset of skypatterns on which to apply the Sky operator. CP+Sky iteratively refines the skyline constraints using the extracted patterns. This leads to better pruning of the search space. Our approach generates the complete set of skypatterns in a generic manner (i.e. with a large set of measures that includes statistical assessments such as the p-value). The practical goal is to make the result of pattern mining useful from a *user-preference point of view*. One strength of the approach lies in the fact that no threshold has to be set, the end-user only needs to specify as input the dataset and the set of measures she is interested in.

An extensive empirical study as well as a case study from chemoinformatics show the efficiency and effectiveness of our two algorithms according to both quantitative and qualitative aspects. Despite the gain in generality brought by the CSP framework and the fact that Aetheris benefits from the pruning strategies based on the anti-monotonicity to extract patterns, CP+Sky competes with Aetheris and even outperforms it in some cases. However, with CP+Sky, the search order of the patterns may significantly impact the efficiency of the CSP solver. Investigating the most beneficial order in which patterns are enumerated is a promising research direction to maximize the effectiveness of the strategy of dynamically posting constraints.

Skypattern mining can generally be applied to a wide range of problems by adapting either the language or the dominance relation. For instance, the language of sequences or graphs can also produce skypatterns. The dominance relation can be changed or extended to take into account different other criteria for user-preferences. We sketch now these two issues.

Language. Our formalization, and especially the skylineability notion, is general enough to be applied to a large set of languages (sequences, trees and graph for instance). However, a change in the language can impact the efficiency of the extraction methods. Regarding Aetheris, the efficiency of the approach is based on Theorem 1 involving the distinct and indistinct operators. As aforementioned, with itemset patterns and the frequency measure, the distinct operator corresponds to the well-known closed or free condensed representations of frequent patterns. Consequently, the efficient extraction of skypatterns for more complex languages (i.e. skyline sequential patterns or skyline graph patterns) is strongly tied to the advances and progress on complex condensed representations of patterns.

Evaluating the distinct operator on more complex patterns efficiently such as sequences, trees and graphs implies additional challenges. To cite one, in the case of sequences, convenient properties such as the *free patterns apriori property* [61], which implies effective search space pruning, cannot be used. However, there are already many methods of extraction dedicated to closed sequential patterns, closed graphs and so on. Extending CP+Sky to other languages is a challenging task due to the difficulty of modeling complex patterns. To the best of our knowledge, only certain types of pattern in sequence mining have been successfully studied [62, 63].

Dominance relation. The dominance relation contains two components: the measures for which basic preferences are expressed (e.g., frequency, area) and the combination of these basic preferences (here, the Pareto composition). Primitive-based measures are flexible enough to allow the user to express a wide variety of criteria. Indeed, classical interestingness measures for pattern mining (such as frequency, growth rate), utility functions and measures of statistical significance (like the p-value) fall within this framework. Of course, Aetheris, as mentioned above, depends again on condensed representations that are well-adapted for the desired measures. Conversely, all interestingness measures are easily expressible with CP+Sky. Through its declarative nature, CP+Sky offers via CSP a very flexible way to change the dominance relationship. For instance the strict dominance (i.e. a pattern is dominated by another pattern when the latter has a better value for all measures in M), is easily configurable with CSP and this relaxation of the dominance relation leads to the mining of soft-skypatterns [2].

Exploratory skypattern mining. We think that the skypattern mining is particularly well suited to exploratory research. Indeed, a strength of our approach is to propose a reduced collection of patterns to the data expert who can quickly analyze it. It would be interesting to integrate the user feedbacks to make skypattern mining more iterative and more exploratory. An interesting avenue is to offer an interactive way to refine the preference criteria by computing the skypattern cube according to all possible subsets of measures [64] and then assist the user with an intuitive navigation. We claim that the skypattern cube exploration will provide a better understanding of the impact of the measures on the problem at hand. Other kinds of interactions are also possible, such as discarding a skypattern to reveal patterns that were previously dominated and could become interesting.

Acknowledgments. This work is partly supported by the ANR (French Research National Agency) funded projects FiCoLoFo ANR-10-BLA-0214 and Hybride ANR-11-BS002-002.

[1] A. Soulet, C. Raïssi, M. Plantevit, B. Crémilleux, Mining dominant patterns in the sky, in: ICDM, IEEE Computer Society, 2011, pp. 655–664.

[2] W. Ugarte, P. Boizumault, S. Loudni, B. Crémilleux, A. Lepailleur, Mining (soft-) skypatterns using dynamic CSP, in: CPAIOR, Vol. 8451 of LNCS, Springer, 2014, pp. 71–87. doi:10.1007/978-3-319-07046-9.

[3] M. J. Zaki, K. Sequeira, Data mining in computational biology, in: S. Aluru (Ed.), Handbook of Computational Molecular Biology, Computer and Information Science Series, Chapman & Hall/CRC Press, 2006, Ch. 38, pp. 1–26.

[4] J. Gasteiger, T. Engel, Chemoinformatics: A Textbook, Weinheim: Wiley VCH, 2003.

[5] L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, X. Lan, Group formation in large social networks: membership, growth, and evolution, in: SIGKDD, ACM, 2006, pp. 44–54.

[6] L. Backstrom, J. M. Kleinberg, R. Kumar, Optimizing web traffic via the media scheduling problem, in: SIGKDD, ACM, 2009, pp. 89–98.

[7] W. Fan, M. Miller, S. J. Stolfo, W. Lee, P. K. Chan, Using artificial anomalies to detect unknown and known network intrusions, in: ICDM, IEEE Computer Society, 2001, pp. 123–130.

[8] R. Agrawal, T. Imielinski, A. N. Swami, Mining association rules between sets of items in large databases, in: SIGMOD, ACM Press, 1993, pp. 207–216.

[9] H. Mannila, H. Toivonen, Levelwise search and borders of theories in knowledge discovery, Data Min. Knowl. Discov. 1 (3) (1997) 241–258.

[10] S. Wrobel, An algorithm for multi-relational discovery of subgroups, in: PKDD, Vol. 1263 of LNCS, Springer, 1997, pp. 78–87.

[11] P. K. Novak, N. Lavrac, G. I. Webb, Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining, Journal of Machine Learning Research 10 (2009) 377–403.

[12] F. Geerts, B. Goethals, T. Mielikäinen, Tiling databases, in: DS, Vol. 3245 of LNCS, Springer, 2004, pp. 278–289.

[13] F. Bonchi, F. Giannotti, C. Lucchese, S. Orlando, R. Perego, R. Trasarti, A constraint-based querying system for exploratory pattern discovery, Inf. Syst. 34 (1) (2009) 3–27.

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

[15] T. Calders, B. Goethals, Non-derivable itemset mining, Data Min. Knowl. Discov. 14 (1) (2007) 171–206.

[16] J. Boulicaut, A. Bykowski, C. Rigotti, Free-sets: A condensed representation of boolean data for the approximation of frequency queries, Data Min. Knowl. Discov. 7 (1) (2003) 5–22.

[17] S. Bistarelli, F. Bonchi, Soft constraint based pattern mining, Data Knowl. Eng. 62 (1) (2007) 118–137.

[18] W. Ugarte, P. Boizumault, S. Loudni, B. Crémilleux, A. Lepailleur, Extracting and summarizing the frequent emerging graph patterns from a dataset of graphs, J. Intell. Inf. Syst. (2013) 1–29.

[19] Y. Ke, J. Cheng, J. X. Yu, Top-k correlative graph mining, in: SDM, SIAM, 2009, pp. 1038–1049. doi:10.1137/1.9781611972795.

[20] J. Wang, J. Han, Y. Lu, P. Tzvetkov, TFP: an efficient algorithm for mining top-k frequent closed itemsets, IEEE Trans. Knowl. Data Eng. 17 (5) (2005) 652–664.

[21] S. Börzsönyi, D. Kossmann, K. Stocker, The skyline operator, in: ICDE, IEEE Computer Society, 2001, pp. 421–430.

[22] J. L. Bentley, H. T. Kung, M. Schkolnick, C. D. Thompson, On the average number of maxima in a set of vectors and applications, J. ACM 25 (4) (1978) 536–543.

[23] T. Calders, C. Rigotti, J. Boulicaut, A survey on condensed representations for frequent sets, in: Constraint-Based Mining and Inductive Databases, Vol. 3848 of LNCS, Springer, 2005, pp. 64–80.

[24] R. T. Ng, L. V. S. Lakshmanan, J. Han, A. Pang, Exploratory mining and pruning optimizations of constrained association rules, in: SIGMOD, 1998, pp. 13–24.

[25] A. Siebes, J. Vreeken, M. van Leeuwen, Item sets that compress, in: SDM, SIAM, 2006, pp. 395–406. doi:10.1137/1.9781611972764.

[26] A. J. Knobbe, E. K. Y. Ho, Pattern teams, in: ECML/PKDD, Vol. 4213 of LNCS, Springer, 2006, pp. 577–584.

[27] L. D. Raedt, A. Zimmermann, Constraint-based pattern set mining, in: SDM, SIAM, 2007, pp. 237–248. doi:10.1137/1.9781611972771.

[28] B. Bringmann, A. Zimmermann, The chosen few: On identifying valuable patterns, in: ICDM, IEEE Computer Society, 2007, pp. 63–72.

[29] G. C. Garriga, P. Kralj, N. Lavrac, Closed sets for labeled data, Journal of Machine Learning Research 9 (2008) 559–580.

[30] K. Kontonasios, T. D. Bie, An information-theoretic approach to finding informative noisy tiles in binary databases, in: SDM, SIAM, 2010, pp. 153–164. doi:10.1137/1.9781611972801.

[31] W. Hämäläinen, M. Nykänen, Efficient discovery of statistically significant association rules, in: ICDM, IEEE Computer Society, 2008, pp. 203–212.

[32] A. Gallo, T. D. Bie, N. Cristianini, MINI: mining informative non-redundant itemsets, in: PKDD, Vol. 4702 of LNCS, Springer, 2007, pp. 438–445.

[33] A. Gionis, H. Mannila, T. Mielikäinen, P. Tsaparas, Assessing data mining results via swap randomization, in: SIGKDD, ACM, 2006, pp. 167–176.

[34] M. Mampaey, N. Tatti, J. Vreeken, Tell me what i need to know: succinctly summarizing data with itemsets, in: SIGKDD, ACM, 2011, pp. 573–581.

[35] T. Guns, S. Nijssen, L. D. Raedt, Itemset mining: A constraint programming perspective, Artif. Intell. 175 (12-13) (2011) 1951–1983.

[36] L. D. Raedt, T. Guns, S. Nijssen, Constraint programming for itemset mining, in: SIGKDD, ACM, 2008, pp. 204–212.

[37] M. Khiari, P. Boizumault, B. Crémilleux, Constraint programming for mining n-ary patterns, in: CP, Vol. 6308 of LNCS, Springer, 2010, pp. 552–567. doi:10.1007/978-3-642-15396-9.

[38] H. T. Kung, F. Luccio, F. P. Preparata, On finding the maxima of a set of vectors, J. ACM 22 (4) (1975) 469–476.

[39] J. Matousek, Computing dominances in e^n, Inf. Process. Lett. 38 (5) (1991) 277–278.

[40] R. E. Steuer, Multiple Criteria Optimization: Theory, Computation and Application, John Wiley, 546 pp, 1986.

[41] D. Papadias, Y. Tao, G. Fu, B. Seeger, Progressive skyline computation in database systems, ACM Trans. Database Syst. 30 (1) (2005) 41–82.

[42] K. Tan, P. Eng, B. C. Ooi, Efficient progressive skyline computation, in: VLDB, Morgan Kaufmann, 2001, pp. 301–310.

[43] A. N. Papadopoulos, A. Lyritsis, Y. Manolopoulos, Skygraph: an algorithm for important subgraph discovery in relational graphs, Data Min. Knowl. Discov. 17 (1) (2008) 57–76.

[44] P. Shelokar, A. Quirin, O. Cordón, Mosubdue: a pareto dominance-based multiobjective subdue algorithm for frequent subgraph mining, Knowl. Inf. Syst. 34 (1) (2013) 75–108.

[45] D. J. Cook, L. B. Holder, Graph-based data mining, IEEE Intelligent Systems 15 (2) (2000) 32–41.

[46] M. van Leeuwen, A. Ukkonen, Discovering skylines of subgroup sets, in: ECML/PKDD, Vol. 8190 of LNCS, Springer, 2013, pp. 272–287. doi:10.1007/978-3-642-40994-3.

[47] B. Négrevergne, A. Dries, T. Guns, S. Nijssen, Dominance programming for itemset mining, in: ICDM, IEEE Computer Society, 2013, pp. 557–566.

[48] F. Pennerath, A. Napoli, The model of most informative patterns and its application to knowledge extraction from graph databases, in: ECML/PKDD, Vol. 5782 of LNCS, Springer, 2009, pp. 205–220. doi:10.1007/978-3-642-04174-7.

[49] A. Soulet, B. Crémilleux, Adequate condensed representations of patterns, Data Min. Knowl. Discov. 17 (1) (2008) 94–110.

[50] A. Soulet, B. Crémilleux, Mining constraint-based patterns using automatic relaxation, Intell. Data Anal. 13 (1) (2009) 109–133.

[51] E. Omiecinski, Alternative interest measures for mining associations in databases, IEEE Trans. Knowl. Data Eng. 15 (1) (2003) 57–69.

[52] R. Dechter, A. Dechter, Belief maintenance in dynamic constraint networks, in: AAAI, AAAI Press / The MIT Press, 1988, pp. 37–42.

[53] G. Verfaillie, N. Jussien, Constraint solving in uncertain and dynamic environments: A survey, Constraints 10 (3) (2005) 253–281.

[54] C. Lecoutre, Constraint Networks: Targeting Simplicity for Techniques and Algorithms, Wiley-ISTE, 2009.

[55] F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006.

[56] A. Lepailleur, G. Poezevara, R. Bureau, Automated detection of structural alerts (chemical fragments) in (eco)toxicology, Comp and Struct Biotech J. (2013) 5:e201302013.

[57] B. Cuissart, G. Poezevara, B. Crémilleux, A. Lepailleur, R. Bureau, Emerging patterns as structural alerts for computational toxicology, in: Contrast Data Mining: Concepts, Algorithms, and Applications, CRC Press, 2013, pp. 269–282.

[58] I. Sushko, E. Salmina, V. Potemkin, G. Poda, I. V. Tetko, Toxalerts: A web server of structural alerts for toxic chemicals and compounds with potential adverse reactions, Journal of Chemical Information and Modeling 52 (8) (2012) 2310–2316.

[59] C. Hansch, S. McCarns, C. Smith, D. Dodittle, Comparative QSAR evidence for a free-radical mechanism of phenol-induced toxicity, Chem. Biol. Interact. 127 (2000) 61–72.

[60] S. Lozano, G. Poezevara, M. Halm-Lemeille, E. Lescot-Fontaine, A. Lepailleur, R. Bissell-Siders, B. Crémilleux, S. Rault, B. Cuissart, R. Bureau, Introduction of jumping fragments in combination with QSARs for the assessment of classification in ecotoxicology, Journal of Chemical Information and Modeling 50 (8) (2010) 1330–1339.

[61] D. Lo, S. Khoo, J. Li, Mining and ranking generators of sequential patterns, in: SDM, SIAM, 2008, pp. 553–564. doi:10.1137/1.9781611972788.

[62] E. Coquery, S. Jabbour, L. Saïs, Y. Salhi, A SAT-based approach for discovering frequent, closed and maximal patterns in a sequence, in: ECAI, Vol. 242 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2012, pp. 258–263.

[63] A. Kemmar, W. Ugarte, S. Loudni, T. Charnois, Y. Lebbah, P. Boizumault, B. Crémilleux, Mining relevant sequence patterns with CP-based framework, in: ICTAI, IEEE Computer Society, 2014, pp. 552–559.

[64] W. Ugarte, P. Boizumault, S. Loudni, B. Crémilleux, Computing skypattern cubes, in: ECAI, Vol. 263 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2014, pp. 903–908.