Computing Skypattern Cubes using Relaxation

Willy Ugarte* Patrice Boizumault*, Samir Loudni* and Bruno Crémilleux*
*GREYC Laboratory (CNRS UMR 6072)
Université de Caen Basse-Normandie, 14032 CAEN, FRANCE
{firstname.lastname}@unicaen.fr

***Abstract*—We propose an effective method to compute the skypattern cubes thanks to a relaxation strategy in the pattern mining process. Our approach is based on the fact that each node of the cube can be approximated by the set of edge-skypatterns (a relaxed form of skypatterns) w.r.t. the whole set of measures M. Then we transform the problem into a skyline cube mining in $|M|$ dimensions. The set of edge-skypatterns can be efficiently mined by using either a dynamic CSP method or an extended version of a static method based on the theoretical relationships between patterns and condensed representations of skypatterns. Experiments conducted on `UCI` datasets and on a real-life dataset (`Mutagenicity`) show the relevance and performance of our approach.**

***Keywords*-Skypattern Cube, Soft Skypattern, Dynamic CSP.**

I. Introduction

The notion of skyline queries [1] has been recently integrated into the pattern discovery paradigm to mine skyline patterns (henceforth called *skypatterns*) [2], [3]. Such queries have attracted considerable attention due to their importance in multi-criteria decision and are usually called *"Pareto efficiency or optimality queries"*. Briefly, given a set of measures, skypatterns are patterns based on a Pareto-dominance relation for which no measure can be improved without degrading the others. As an example, a user selecting a set of patterns may prefer a pattern with a high frequency, large length and a high confidence. In this case, a pattern x_i dominates another pattern x_j if $freq(x_j) \geq freq(x_i)$, $size(x_j) \geq size(x_i)$, $confidence(x_j) \geq confidence(x_i)$ where at least one strict inequality holds. Given a set of patterns, the skypattern set contains the patterns that are not dominated by any other pattern. Skypatterns are highly interesting because they do not require any threshold on the measures and the dominance relation gives a global interest with semantics easily understood by the user.

Up to now, skypatterns are computed according to a fixed number of measures. In practice, users do not know the exact role of each measure and it is difficult to beforehand select the most appropriate set of measures. Ideally, users would like to keep all the measures potentially useful, look what happens on a skypattern set by removing or adding a measure to evaluate the impact of measures and then converge to a proper skypattern set. Similarly to the notion of the skyline cube in the database [4], users would like to have available on line the *skypattern cube*. Each element of the cube is a *node* which associates to a subset of the measures its skypattern set. By comparing two neighboring nodes, which are differentiated by adding or removing one measure, users can observe the new skypatterns and the ones which die out. It greatly helps to better understand the role of the measures. Moreover, users can spot that different subsets of measures have the same skypattern set: such an equivalence class over subsets of measures shows useless measures (i.e., measures that can be added to a set of measures without changing the skypattern set). To sum up, the cube is the proper structure to enable various user queries in an efficient manner and to discover the most interesting skypattern sets. Therefore the problem of efficient computing of the skypattern cube is the focus of this paper.

More formally, given a set M of n measures, the skypattern cube has 2^n-1 possible non-empty skypattern subsets. All these subsets should be precomputed to efficiently handle various queries of users. An obvious and naive method needs the computing of the 2^n-1 skypattern sets leading to a prohibitive cost.

Very recently, [5] has designed the first (and unique) approach to compute skypattern cubes. The key idea of this bottom-up approach is to automatically collect on a parent node the skypatterns which can be derived from its child nodes (if k measures are associated to a parent node, its child nodes are the nodes defined by the $\binom{k}{k-1}$ subsets of $(k-1)$ measures). Other skypatterns are computed on the fly. Independently, soft skypatterns have been recently introduced by [3]. As the skypatterns suffer from the stringent aspect of the constraint-based pattern framework, soft skypatterns enable to capture valuable patterns occurring in the dominated area. [3] has proposed two kinds of soft skypatterns: the *edge-skypatterns* that belong to the Pareto frontier (while skypatterns are vertices of this frontier), and the *δ-skypatterns* that are close to the boundary.

This paper revisits in depth the skypattern cube problem by proposing a new and effective method to compute the skypattern cubes thanks to a relaxation strategy in the pattern mining process. Our approach consists first in demonstrating that every node of the cube is included in the set of the edge-skypatterns w.r.t. the whole set of measures M. Then we transform the problem into a skyline cube mining problem (in $|M|$ dimensions) for which several extractors have been already developed [4], [6]. We use two alternative methods for efficiently mining the set of soft skypatterns: a dynamic CSP (Constraint Satisfaction Problem) method [3] and an extension of a static method based on the theoretical relationships between pattern and condensed representations of skypatterns [2]). Experiments conducted on UCI datasets and on a real-life dataset (Mutagenicity) show that our relaxation based approach clearly outperforms the bottom-up approach [5], and that the approximation we performed is of very good quality.

The paper is organized as follows. Section II defines the (soft) skypatterns and describes the bottom-up method used to compute the skypattern cubes. Section III provides an overview on skyline cubes. Section IV presents our contribution and Section V is devoted to experimentations.

II. Context and Definitions

A. Context

Let $\mathcal{I}$ be a set of distinct literals called *items*. An itemset (or pattern) is a non-null subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L}_{\mathcal{I}} = 2^{\mathcal{I}} \backslash \emptyset$. A transactional dataset $\mathcal{T}$ is a multiset of patterns of $\mathcal{L}_{\mathcal{I}}$. Each pattern (or transaction) is a database entry. Fig. 1a presents a transactional dataset $\mathcal{T}$ where each transaction t_i gathers articles described by items denoted $A,\ldots,F$. The traditional example is a supermarket database in which each transaction corresponds to a customer and every item in the transaction to a product bought by the customer. An attribute (*price*) is associated to each product (see Fig. 1a).

Constraint-based pattern mining aims at extracting all patterns x of $\mathcal{L}_{\mathcal{I}}$ satisfying a query $q(x)$ (conjunction of constraints) which is usually called *theory* [7]: $Th(q) = \{x \in \mathcal{L}_{\mathcal{I}} \mid q(x)\ is\ true\}$. A common example is the frequency measure leading to the minimal frequency constraint ($freq(x) \geq \theta$). The latter provides patterns x having a number of occurrences in the dataset exceeding a given minimal threshold θ. There are other usual measures for a pattern x:

- *size*(x) is the number of items that pattern x contains.
- *area*$(x) = freq(x) \times size(x)$.
- *min*$(x.att)$ (resp. *max*$(x.att)$) is the smallest (resp. highest) value of the item values of x for attribute att.
- *mean*$(x) = \frac{(min(x.att)+max(x.att))}{2}$.

In many applications, it is highly appropriated to look for contrasts between subsets of transactions The growth-rate is a well-used contrast measure highlighting patterns whose frequency increases significantly from one subset to another (See Section V-B).

Definition 1 (Growth-rate)**.** *Let $\mathcal{T}$ be a database partitioned into two subsets $\mathcal{D}_1$ and $\mathcal{D}_2$. The growth-rate of a pattern x from $\mathcal{D}_2$ to $\mathcal{D}_1$ is:*

$$m_{gr}(x) = \frac{|\mathcal{D}_2| \times freq(x, \mathcal{D}_1)}{|\mathcal{D}_1| \times freq(x, \mathcal{D}_2)}$$

The collection of patterns contains redundancy w.r.t. measures. Given a measure m, two patterns x_i and x_j are said to be equivalent if $m(x_i) = m(x_j)$. A set of equivalent patterns forms an equivalence class w.r.t. m. The largest element (i.e. the one with the highest number of items) of an equivalence class is called a *closed pattern*. The set of closed patterns is a compact representation of the patterns (i.e we can derive all the patterns with their exact value for m from the closed ones). This definition is straightforwardly extended to a set of measures M.

B. Skypatterns

Skypatterns have been recently introduced by [2]. Such patterns enable to express a user-preference point of view w.r.t. a dominance relation. Let M be a set of measures.

Definition 2 (Pareto Dominance)**.** *A pattern x_i dominates another pattern x_j w.r.t. M (denoted by $x_i \succ_M x_j$), iff $\forall m \in M, m(x_i) \geq m(x_j)$ and $\exists m \in M, m(x_i) > m(x_j)$.*

Definition 3 (Skypattern operator)**.** *A skypattern w.r.t. M is a pattern not dominated w.r.t. M. The skypattern operator $Sky(M)$ returns all the skypatterns w.r.t. M:*

$$Sky(M) = \{x_i \in \mathcal{L}_{\mathcal{I}} \mid \nexists x_j \in \mathcal{L}_{\mathcal{I}}, x_j \succ_M x_i\}$$

Example 1. *C, D and CD, are skypatterns w.r.t. M=$\{m_1$, $m_2\}$ since they are not dominated by any other pattern. BCE in not a skypattern w.r.t. M since $CD \succ_M CDE$.*

Two patterns x_i and x_j are *indistinct* w.r.t. M (denoted by $x_i =_M x_j$) iff $\forall m \in M, m(x_i) = m(x_j)$. Two patterns x_i and x_j are *incomparable* w.r.t. M (denoted by $x_i \prec\succ_M x_j$) iff $(x_i \not\succ_M x_j)$ and $(x_j \not\succ_M x_i)$ and $(x_i \neq_M x_j)$.

Definition 4 (Incomparable skypattern)**.** *A pattern $x \in Sky(M)$ is incomparable w.r.t M iff $\forall x_i \in Sky(M)$ s.t. $x_i \neq x$, $x_i \prec\succ_M x$.*

Example 2. *Pattern CD is an incomparable skypattern w.r.t. M=$\{m_1$, $m_2\}$.*

Definition 5 (Indistinct skypattern)**.** *A pattern $x \in Sky(M)$ is indistinct w.r.t. M iff $\exists x_i \neq x \in Sky(M)$ s.t. $(x_i =_M x)$.*

It is easy to see that incomparable skypatterns and indistinct ones w.r.t. M constitute a partition of $Sky(M)$. Moreover, $=_M$ is an equivalence relation (i.e., the relation is reflexive, symmetric and transitive). So, indistinct skypatterns can be gathered into a group.

Definition 6 (Indistinct Skypattern Group (ISG))**.** *$S \subseteq Sky(M)$ is an indistinct skypattern group w.r.t. M, iff (i) $|S| \geq 2$, (ii) $\forall x_i$,$x_j$$\in$ S, $(x_i =_M x_j)$ and (iii) $\forall x_i \in S$, $\forall x_j \in Sky(M) \setminus S, (x_i \prec\succ_M x_j)$.*

Example 3. *$\{C$, $D\}$ is an ISG w.r.t. M=$\{m_1$, $m_2\}$ since both C and D are skypatterns w.r.t. M and $m_1(C) = m_1(D)$ and $m_2(C) = m_2(D)$.*

Trans.	Items							
t_1	c_1			B	C		E	F
t_2	c_1			B	C	D	E	
t_3	c_1				C	D		
t_4	c_1				C	D		F
t_5		c_2	A			D		F
t_6		c_2	A	B	C	D	E	F
t_7		c_2	A		C	D		F

Item	A	B	C	D	E	F
Price	10	20	70	40	50	60

(a) Transactional dataset $\mathcal{T}$.

Subset of M	**Skypattern set**
$\{m_1, m_2, m_3, m_4\}$	{BCE, C, CD, CF}
$\{m_1, m_2, m_3\}$	{C, CD, CF}
$\{m_1, m_2, m_4\}$	{BCE, C, CD}
$\{m_1, m_3, m_4\}$	{C}
$\{m_2, m_3, m_4\}$	{BCE, C, CD, CF}
$\{m_1, m_2\}$	{C, CD, D}
$\{m_1, m_3\}$	{C}
$\{m_1, m_4\}$	{C}
$\{m_2, m_3\}$	{C, CD, CF}
$\{m_2, m_4\}$	{BCE, CD}
$\{m_3, m_4\}$	{C}
$\{m_1\}$	{C, D}
$\{m_2\}$	{CD}
$\{m_3\}$	{C}
$\{m_4\}$	{B, BE, BCE, C, CE, E}

(b) Skypattern cube for M.

Pattern	m_1	m_2	m_3	m_4
B	3	3	20	1.500
BCE	3	9	44	1.500
BE	3	6	35	1.500
C	6	6	70	1.500
CD	5	10	55	1.125
CDE	2	6	55	0.750
CDF	3	9	55	0.375
CE	3	6	60	1.500
CEF	2	6	60	0.750
CF	4	8	64	0.750
D	6	6	40	0.750
E	3	3	50	1.500

(c) Edge-skypatterns for M.

FIGURE 1: M=$\{m_1 : freq, m_2 : area, m_3 : mean, m_4 : growth\text{-}rate\}$.

Two methods have been proposed for mining skypatterns:
- `Aetheris` [2] takes benefit of theoretical relationships between pattern condensed representations and skypatterns. `Aetheris` proceeds in two steps : first, condensed representations of the whole set of patterns (i.e. closed patterns according to the considered set of measures) are extracted; then, the sky operator (see Definition 3) is applied.
- `CP+SKY` [3] mines skypatterns using Dynamic CSP (see Section IV-B2). Finally, experiments performed by [3] show that both methods are equally effective.

C. Edge-skypatterns

Soft skypatterns have been recently introduced by [3]. As the skypatterns suffer from the stringent aspect of the constraint-based pattern framework, soft skypatterns enable to capture valuable patterns occurring in the dominated area. [3] has proposed two kinds of soft skypatterns: the *edge-skypatterns* that belongs to the edge of the dominance area and the *δ-skypatterns* that are close to this edge.

The key idea is to strengthen the dominance relation in order to soften the notion of non dominated patterns. Edge-skypatterns are also defined according to a dominance relation and a Sky operator. Let M be a set of measures.

Definition 7 (Strict Dominance)**.** *A pattern x_i strictly dominates a pattern x_j w.r.t M (denoted by $x_i \gg_M x_j$), iff $\forall m \in M$, $m(x_i) > m(x_j)$.*

Definition 8 (Edge operator)**.** *An edge-skypattern w.r.t. M is a pattern not strictly dominated w.r.t. M. The operator $Edge\text{-}Sky(M)$ returns all the edge-skypatterns w.r.t. M:*

$$Edge\text{-}Sky(M) = \{x_i \in \mathcal{L_I} \mid \nexists x_j \in \mathcal{L_I}, x_j \gg_M x_i\}$$

Edge-skypatterns belong to the Pareto frontier while skypatterns are vertices of this frontier: every skypattern is an edge-skypattern.

Theorem 1. *$Sky(M) \subseteq Edge\text{-}Sky(M)$.*

Proof: let $x_i, x_j \in \mathcal{L_I}$, if $x_i \gg_M x_j$ then $x_i \succ_M x_j$. So, $Sky(M) \subseteq Edge\text{-}Sky(M)$. ■

Example 4. *Patterns C, CD and CF are (incomparable) skypatterns w.r.t. M=$\{m_2, m_3\}$. $Edge\text{-}Sky(M)$ = $\{C, CD, CF, CDF\}$ since pattern CDF is also an edge-skypattern (CDF is not strictly dominated w.r.t. M by any other pattern).*

D. Skypattern cube

The skypattern cube over a set of measures M consists in all the $2^{|M|}-1$ skypattern sets $Sky(M_u)$ for any non-empty subset $M_u \subseteq M$.

Definition 9 (Skypattern Cube)**.** *Let M be a set of measures.*

$$SkyCube(M) = \{(M_u, Sky(M_u)) \mid M_u \subseteq M, M_u \neq \emptyset\}$$

As different subsets of measures may lead to a same skypattern set, a concise representation of the cube can be provided, without loss of information, by defining an equivalence relation over subsets of measures having the same skypattern set:

Definition 10 (Equivalence between sets of measures)**.** *Let M_u and M_v two sets of measures. M_u and M_v are said to be equivalent iff $Sky(M_u) = Sky(M_v)$.*

Example 5. *Fig. 1b depicts the skypattern cube w.r.t. M by associating, to each of the 2^4-1 nodes, its skypattern* *set. There are 9 classes of equivalence for the concise representation.* $\{m_3\}$, $\{m_1, m_3\}$, $\{m_1, m_4\}$, $\{m_3, m_4\}$ *and* $\{m_1, m_3, m_4\}$ *belong to the same class of equivalence since they all have the same skypattern set, i.e.* $\{C\}$*. However, the class of equivalence for* $\{m_1, m_2, m_4\}$ *is a singleton.*

E. Computing skypattern cubes

The first (and unique) approach to compute skypattern cubes has been proposed by [5]. CP+SKY+CUBE is a bottom-up approach that relies on two derivation rules collecting skypatterns of a parent node from its child nodes without any dominance test.

Two theorems (see [5] for their proof) define the derivation rules that enable to derive a subset of skypatterns of a parent node. Theorem 2 states that all the incomparable skypatterns of a child node remain incomparable skypatterns in its parent nodes. Theorem 3 exhibits the indistinct skypatterns of a child node that remain skypatterns in its parent nodes. Moreover, if a skypattern in a parent node is also a skypattern in at least one of its child nodes, then it will be necessary collected by one of these rules.

Theorem 2 (Incomparability Rule)**.** *Let* $M_u \subseteq M$*. If* x *is an incomparable skypattern w.r.t.* M_u *then* $\forall m \in M \setminus M_u$, $x \in Sky(M_u \cup \{m\})$*. Moreover* x *is incomparable w.r.t.* $M_u \cup \{m\}$.

Theorem 3 (ISG Rule)**.** *Let* $M_u \subseteq M$ *and* S *an ISG w.r.t.* M_u*.* $\forall m \in M \setminus M_u$*, each skypattern* $x \in S$ *such that* $m(x) = \max_{x_i \in S}\{m(x_i)\}$ *is a skypattern w.r.t.* $M_u \cup \{m\}$.

Non-derivable skypatterns are computed on the fly thanks to Dynamic CSP. The bottom-up principle enables to provide a concise representation of the cube based on skypattern equivalence classes without any supplementary effort.

III. Related Work

Mining skypatterns is far different from mining skylines. Skyline queries focus on the extraction of tuples of the dataset and assume that all skylines belong to the dataset [1]. The skypattern mining task consists in extracting patterns which are elements of the frontier defined by the given measures [2], [3], [8], [12]. The skypattern problem is clearly harder because the search space for skypatterns ($O(2^{|\mathcal{I}|})$) is much larger than the search space for skylines ($O(|\mathcal{T}|)$).
Computing Skyline Cubes. Several strategies to share skyline computation in different nodes have been proposed: [9], [10] but they have to cope with the problem of enumerating skylines over all possible nodes. In [6], [11], skyline groups have been introduced as an alternative to skycube structure. This can be shown as a way to identify the semantics of skyline points. Recently, Orion [4] optimized skyline group computation using skyline derivation rules and closure operators between subspace skylines. Finally, a *group-by skyline cube* [13] was introduced as an interesting extension of the skycube by combining group-by operation. All of these techniques address only skylines.

IV. Computing skypattern cubes by relaxation

This section presents a new method to compute skypattern cubes thanks to a relaxation strategy in the pattern mining process. Our approach is based on the fact that each node of the cube can be approximated by $Edge\text{-}Sky(M)$, the set of edge-skypatterns w.r.t. the whole set of measures M. Then we convert the problem into a skyline cube mining in $|M|$ dimensions to process it more efficiently.

Section IV-A provides the ”why and how” of our approximation-based approach. Section IV-B describes how $Edge\text{-}Sky(M)$ can be efficiently mined by using either a dynamic CSP method or an extended version of Aetheris. Finally, Section IV-C shows how the computation of the skypattern cube (according to M) can then be converted to the computation of a skyline cube (in $|M|$ dimensions).

A. Approximating each node by $Edge\text{-}Sky(M)$

On one side, [5] has proposed a bottom-up approach that relies on two derivation rules collecting the skypatterns of a parent node from its child nodes without any dominance test (see Section II-E). The first derivation rule deals with incomparable skypatterns (see Theorem 2) while the second one is devoted to indistinct skypatterns (see Theorem 3). On the other side, [3] has introduced edge-skypatterns (see Section II-C). Although these notions have been separately introduced, they are closely linked as shown below.

1) Key idea: Contrary to the Sky operator, the $Edge\text{-}Sky$ operator is monotonic (see Theorem 5). As a consequence, each node of the cube is included in $Edge\text{-}Sky(M)$ (see Theorem 6). This is the greatest outcome of the paper: the unique set $Edge\text{-}Sky(M)$ is a superset of all the nodes of the skypattern cube w.r.t. M.

Example 6. *The various elements of* $Edge\text{-}Sky(M)$ *are reported in Column 1 of Fig. 1c.* $Edge\text{-}Sky(M)$ *is a superset of each skypattern set of the cube (see Fig. 1b).*

2) Progression: Let $M_u \subseteq M$ and $m \in M \backslash M_u$, Theorem 4 exhibits a particular kind of patterns, namely patterns that are skypatterns for a child node but are not skypatterns for a father node. But, as they are edge-skypatterns for the child node (see Theorem 1), there will also be edge-skypatterns for the father node. Theorem 5 proves that the $Edge\text{-}Sky$ operator is monotonic, and Theorem 6 provides the final result.

Theorem 4. *Let* $M_u \subseteq M$*,* $m \in M \backslash M_u$*, and* S *an ISG w.r.t.* M_u*. Let* $S' = \{x \in S \mid m(x) = \max_{x_i \in S}\{m(x_i)\}$*.*
If S' *is a singleton then the unique skypattern is incomparable w.r.t.* $M_u \cup \{m\}$ *else* S' *is an ISG w.r.t.* $M_u \cup \{m\}$*. Finally all* $x \in S \setminus S'$ *are not skypatterns for* $M_u \cup \{m\}$.

Proof: If the maximum is unique, then the pattern is incomparable w.r.t. $M_u \cup \{m\}$. If not, all these patterns are indistinct w.r.t. $M_u \cup \{m\}$ since they all have the same value for m and for every $m' \in M_u$ (see Theorem 3). Finally, let $x \in S \setminus S'$. x cannot be a skypattern w.r.t. $M_u \cup \{m\}$ since every $x' \in S'$ dominates x because $m(x') > m(x)$ and $\forall m' \in M_u, m'(x') = m'(x)$. ∎

Theorem 5 (monotonicity of *Edge-Sky*)**.** *Let M be a set of measures, and $M_u \subseteq M$. Then $\forall m \in M \setminus M_u$, Edge-Sky$(M_u) \subseteq$ Edge-Sky$(M_u \cup \{m\})$.*

By contradiction: Let $x \in$ *Edge-Sky*(M_u) and assume that $x \notin$ *Edge-Sky*$(M_u \cup \{m\})$. So, $\exists y$ s.t. $y \gg_{M_u \cup \{m\}} x$. We deduce: (1) $\forall m_i \in M_u, m_i(y) > m_i(x)$ and (2) $m(y) > m(x)$. (1) contradicts that $x \in$ *Edge-Sky*(M_u). ∎

Theorem 6 (Fundamental Result)**.** *Let M be a set of measures. $\forall M_u \subseteq M$, Sky$(M_u) \subseteq$ Edge-Sky(M).*

Proof: *Sky*$(M_u) \subseteq$ *Edge-Sky*(M_u) (see Theorem 1). Using Theorem 5, *Edge-Sky*$(M_u) \subseteq$ *Edge-Sky*(M). So, $\forall M_u \subseteq M$, *Sky*$(M_u) \subseteq$ *Edge-Sky*(M). ∎

B. Computing Edge-Sky(M)

1) Using `Edge-Aetheris`: We have built an extension of `Aetheris` in order to compute edge-skypatterns. Like `Aetheris` (see Section II-B), `Edge-Aetheris` proceeds in two steps: first, the pattern condensed representation made of the whole set of closed patterns are extracted; then, the *Edge-Sky* operator (see Definition 8) is applied. The first step is the same as for `Aetheris`, while the second one has been performed by implementing the strict dominance relation as well as the *Edge-Sky* operator.

2) Using Dynamic CSP [3]: The main idea is to improve the mining step during the process thanks to constraints dynamically posted and stemming from the current set of the candidate skypatterns. This process stops when the dominated area cannot be enlarged. The completeness of our approach is insured by the completeness of the CSP solver.

A Dynamic CSP [14] is a sequence $P_1, P_2, ..., P_n$ of CSP, each one resulting from some changes in the definition of the previous one. Each time a new solution is found, new constraints are added. Such constraints will survive backtracking and state that next solutions should verify both the current set of constraints and the added ones.

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

- $q_1(x) = closed_M(x)$
- $q_{i+1}(x) = q_i(x) \wedge \neg(s_i \gg_M x)$ where s_i is the first solution to query $q_i(x)$

First, the constraint $closed_M(x)$ states that x must be a closed pattern, it allows to reduce the number of redundant patterns (see Section II-A). Then, the added constraint $\neg(s_i \gg_M x)$ states that the next solution (which is searched) will not be strictly dominated by s_i (see Definition 7).

Each time the first solution s_i to query $q_i(x)$ is found, a new constraint $\neg(s_i \gg_M x) \equiv \vee_{m \in M}\, m(s_i) \le m(x)$ is dynamically posted leading to reduce the search space. This process stops when the dominated area cannot be enlarged (i.e. there exits n s.t. query $q_{n+1}(x)$ has no solution).

C. Computing the skypattern cube

1) From one cube to another: This section shows how the problem of computing a skypattern cube w.r.t. a set of measures M can be converted into an equivalent problem of computing a skyline cube in $|M|$ dimensions.

Let M be a set of measures and k=$|M|$. Let f be a mapping from $\mathcal{L}_{\mathcal{I}}$ to $\mathbb{R}^k$ that associates, to each pattern $p \in \mathcal{L}_{\mathcal{I}}$, a data point $f(p) \in \mathbb{R}^k$ with coordinates $(m_1(p), m_2(p), \ldots, m_k(p))$. Let $P = \{f(p) \mid p \in \mathcal{L}_{\mathcal{I}}\}$. P is a multiset: let p_i and p_j s.t. $p_i \neq p_j$. If p_i and p_j are indistinct w.r.t. M then $f(p_i) = f(p_j)$.

Example 7. *Fig. 1c reports the mapping between Edge-Sky(M) and data points of $\mathbb{R}^4$ ($|M| = 4$). $f(B)$ (resp. $f(BCE)$) is the data point with coordinates* $(3, 3, 20, 1.500)$ *(resp.* $(3, 9, 44, 1.500)$*).*

Let $M_u \subseteq M$ and *Skyline*(M_u) be the set of skyline points (of P) w.r.t. M_u. Then we have the following property, whose proof is immediate reasoning by contradiction:

Theorem 7. *Let M be a set of measures. $\forall M_u \subseteq M$, Sky$(M_u) = \{p \in \mathcal{L}_{\mathcal{I}} \mid f(p) \in$ Skyline$(M_u)\}$.*

Consequently, the equivalence classes for skypatterns (i.e p) can be deduced from the equivalence classes for skylines (i.e $f(p)$) directly to obtain a concise representation of the skypattern cube.

Example 8. *Let M_u=$\{m_2, m_3\}$. Using the skyline extractor and the mapping f, we obtain Skyline(M_u) =* $\{(6, 6, 70, 1.500), (5, 10, 55, 1.125), (4, 8, 64, 0.750)\}$. *We can deduce that Sky(M_u) = $\{C, CD, CF\}$ using the mapping f (see Fig. 1c).*

2) Practical use: Applying a skyline extractor to $f(\mathcal{L}_{\mathcal{I}})$ would constitute a naive approach since $f(\mathcal{L}_{\mathcal{I}})$ contains $2^{|\mathcal{I}|}$ points. Let E be a superset of all the skypatterns. Then applying a skyline cube computation method on $f(E)$ ensures to provide the skypattern cube (see Theorem 7). We use E = *Edge-Sky*(M) which is in practice a good superset of all the skypatterns (see Section V-B). Indeed, edge-skypatterns belong to the Pareto frontier while skypatterns are vertices of this frontier. Another way should consider the closed patterns but their number is too large (cf. Section V).

V. Experiments

This section wants to assess two points: the CPU times and the quality of our approximation using *Edge-Sky*(M).

Experiments were conducted on 2 types of datasets: a real-life dataset Mutagenicity (see Section V-B) and several UCI datasets (see Section V-C). Experiments show that our relaxation based approach clearly outperforms the bottom-up approach CP+SKY+CUBE [5], and that the approximation we performed is of very good quality.

A. Experimental protocol

We used MICMAC [15] to mine closed patterns, and Orion[1] [4] to compute skyline cubes, since Orion is one of the most efficient skyline extractor and provides the concise representation of a skyline cube.

1) CPU-time analysis: let M be a set of measures. We compare six methods:

- two base-line methods:
  - Base-Line-Aetheris applies Aetheris to each non empty subset of M,
  - Base-Line-CP+SKY applies CP+SKY to each non empty subset of M,
- the bottom-up approach: CP+SKY+CUBE (see Section II-E).
- three approximation based methods:
  - MICMAC+Orion mines the closed patterns using MICMAC and then applies Orion,
  - Edge-Aetheris+Orion computes *Edge-Sky*(M) using Edge-Aetheris and then applies Orion (see Section IV-B1),
  - CP+Edge-SKY+Orion computes *Edge-Sky*(M) using CP+Edge-SKY and then applies Orion (See Section IV-B2).

For the two base-line methods, reported CPU-time is the sum of CPU-times required for each non-empty subset of M. For the three approximation-based ones, reported CPU-time is the sum of CPU-times of the two steps: first, computing the approximation (either closed patterns or edge-skypatterns), and then computing the cube using Orion.

All experiments were conducted on a computer running Linux with a core i3 processor at 2.13 GHz.

2) Effectiveness of the approximation: we consider, for a set of measures M:

- the number of (distincts) skypatterns of the cube:
  $nCube(M) = | \cup_{M_u \subseteq M, M_u \neq \emptyset} \; Sky(M_u) \; |$,
- the number of closed pattern w.r.t. M: $nClosed(M)$,
- the number of edge-skypatterns w.r.t. M:
  $nEdge(M) = |Edge\text{-}Sky(M)|$.

To evaluate the effectiveness of our approximation, we determine the ”extra” mined patterns by the selected approach. For MICMAC+Orion the ”extra” are quantified by the proportion of closed patterns that are not skypatterns (for any node of the cube). For Edge-Aetheris+Orion and CP+Edge-SKY+Orion, the ”extra” are quantified by the proportion of edge-skypatterns that are not skypatterns (for any node of the cube).

B. Skypattern cubes for Mutagenicity dataset

This section reports an experimental evaluation on a real-life dataset of large size extracted from mutagenicity data [16] (a major problem in risk assessment of chemicals). This dataset has $|\mathcal{T}|$=$6,512$ transactions encoding chemicals and $|\mathcal{I}|$=$1,073$ items[2] encoding frequent closed subgraphs previously extracted from $\mathcal{T}$ with a 2% relative frequency threshold. Chemists use up to $|M|$=11 measures, five of them are typically used in contrast mining (frequency and growth-rate) and enable to express different kinds of background knowledge. The other six measures are related to topological, geometrical and chemical properties.

1) CPU-time analysis: Fig. 2 compares the CPU-times of the six methods according to the number of measures. The y-scale is logarithmic. Table I further explores the CPU-times. For each method, and for $|M|$=k, the reported CPU-time is the average of CPU-times over all $\binom{11}{k}$ possible skypattern cubes.

The two base-line methods have a similar behavior since Aetheris and CP+SKY are equally effective (see Section II-B). As expected, base-line methods are very far from the other four methods.

The method based on the approximation by the closed patterns (MICMAC+Orion) is of very average quality. Indeed, approximating by closed patterns is too coarse compared with approximating by edge-patterns, and generates a huge number of data points (see Section V-B2).

The two methods based on the approximation by *Edge-Sky*(M) are equally effective and clearly outperform the bottom-up approach CP+SKY+CUBE. Moreover, the greater the number of measures, the greater the speedup. Whenever a new measure is added, the number of nodes to consider is twice bigger and the speed-ups are multiplied by a factor of 1.25 to 1.45 (see Columns 7 and 8 of Table I).

Finally, we measured the CPU times required to compute *Edge-Sky*(M) and the CPU times spent by Orion. For $8 \leq |M| \leq 11$, computing *Edge-Sky*(M) represents 90% of the total CPU times. Once again, it shows the importance of the quality of the approximation. The repartition seems to be disproportionate, but computing skypattern cubes is much harder than computing skyline cubes (see Section III).

2) Effectiveness of the approximation: To asses the quality of the approximation of a skypattern cube by *Edge-Sky*(M), Table II reports, for different values of $|M|$, the ratio of edge-skypatterns (resp. closed patterns) that are not skypatterns in the cube. Column 1 corresponds to the number of measures. Column 2 indicates the total number of (distinct) skypatterns of the cube. Column 3 reports the number of edge-skypatterns. Column 4 gives the number of

[1] https://github.com/leander256/Orion

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

	CPU-Times						Speed-Ups			
							(7)		(8)	
$\lvert M\rvert$	(1)	(2)	(3)	(4)	(5)	(6)	$\frac{(3)}{(5)}$	$\frac{(4)}{(5)}$	$\frac{(3)}{(6)}$	$\frac{(4)}{(6)}$
2	15m:42s	17m:27s	14m:39s	7m:03s	6m:41s	**5m:01s**	2.19	1.05	2.91	1.40
3	50m:19s	47m:55s	28m:11s	18m:44s	10m:13s	**8m:33s**	2.76	1.83	3.30	2.19
4	2h:08m:40s	1h:56m:58s	1h:00m:44s	48m:43s	**16m:47s**	19m:06s	3.62	2.90	3.18	2.55
5	5h:21m:47s	4h:28m:09s	1h:37m:39s	1h:19m:30s	29m:42s	**24m:04s**	3.29	2.68	4.06	3.30
6	10h:49m:45s	9h:50m:41s	3h:29m:20s	2h:04m:45s	**30m:06s**	32m:28s	6.95	4.14	6.44	3.84
7	19h:01m:22s	21h:08m:11s	7h:20m:13s	3h:09m:34s	37m:37s	**34m:14s**	11.70	5.04	12.86	5.54
8	58h:05m:32s	44h:41m:11s	15h:15m:33s	4h:40m:02s	39m:14s	**35m:21s**	23.33	7.13	25.90	7.92
9	131h:03m:16s	93h:36m:37s	31h:31m:17s	6h:43m:06s	39m:44s	**35m:45s**	47.60	10.14	52.89	11.27
10	175h:17m:57s	194h:46m:36s	64h:44m:06s	9h:26m:41s	41m:25s	**36m:24s**	93.78	13.68	106.66	15.56
11	523h:11m:58s	402h:27m:40s	131h:58m:47s	12h:59m:35s	42m:07s	**39m:52s**	187.98	18.51	198.63	19.56

(1) Base-Line-Aetheris (3) MICMAC+Orion (5) Edge-Aetheris+Orion
(2) Base-Line-CP+SKY (4) CP+SKY+CUBE (6) CP+Edge-SKY+Orion

TABLE I: COMPARING CPU-TIMES (MUTAGENICITY).

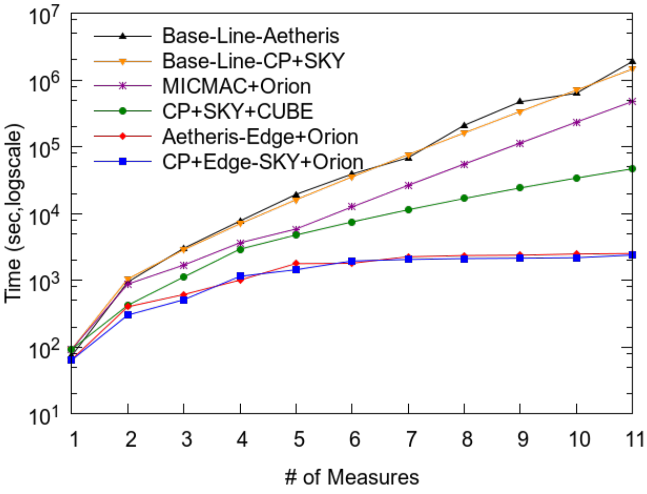

FIGURE 2: COMPARING CPU-TIMES (MUTAGENICITY).

closed patterns. Column 5 (resp. 6) denotes the ratio of edge-skypatterns (resp. closed patterns) that are not skypatterns. For $|M|{=}k$, reported values are the average values over all $\binom{11}{k}$ possible skypattern cubes.

Columns 5 and 6 clearly show that $\mathit{Edge\text{-}Sky}(M)$ provides a much better approximation than $\mathit{Closed}(M)$: i) the number of edge-skyppatens is always smaller than the number of closed patterns, ii) for $|M| \geq 6$, the ratio for edge-skypatterns is between 30% and 50%, while for closed patterns this ratio is always greater than 75%.

C. Skypattern cubes for UCI datasets

Experiments were carried out on 14 various (in terms of dimensions and density) datasets from UCI[3] benchmarks. We considered 5 measures $M{=}\{\mathit{freq}, \mathit{max}, \mathit{area}, \mathit{mean}, \mathit{growth\text{-}rate}\}$. Measures using numeric values, like mean, were applied on attribute values that were randomly generated within the range $[0..1]$.

$\lvert M\rvert$	$nCube(M)$	$nEdge(M)$	$nClosed(M)$	(5)	(6)
2	158.84	740.56	311,254.41	0.79	0.99
3	317.96	1,263.55	344,641.68	0.75	0.99
4	588.38	2,056.70	362,054.86	0.71	0.99
5	1,454.11	3,497.94	371,127.89	0.58	0.99
6	3,507.12	6,513.64	382,273.50	0.46	0.99
7	8,352.34	13,316.40	394,456.48	0.37	0.97
8	19,537.90	29,079.40	412,000.00	0.33	0.95
9	44,552.70	65,687.70	437,734.00	0.32	0.89
10	98,515.50	150,060.00	455,986.55	0.34	0.78
11	110,689.00	172,447.00	483,320.00	0.36	0.77

(5) $1 - \frac{nCube(M)}{nEdge(M)}$ (6) $1 - \frac{nCube(M)}{nClosed(M)}$

TABLE II: EFFECTIVENESS OF THE APPROXIMATION (MUTAGENICITY).

1) CPU-time analysis: Table III compares the CPU-times for computing $\mathit{Sky}(M)$ for the six methods on every dataset. As for the Mutagenicity dataset: i) the two baseline methods have a similar behavior but are far from the other four methods, ii) the two methods based on the approximation by $\mathit{Edge\text{-}Sky}(M)$ are equally effective and outperform the bottom-up approach CP+SKY+CUBE, iii) MICMAC+Orion is of average quality, except for a single dataset (mushroom) where MICMAC+Orion is the most efficient method. This is due to the low density of mushroom (about 19%) and to the small number of closed patterns w.r.t its size. Finally, the speed-ups are lower than those of Mutagenicity since $|M|$ is small.

2) Effectiveness of the approximation: Column 5 (resp. Column 6) of Table IV reports, for each dataset, the ratio of edge-skypatterns (resp. closed patterns) that are not skypatterns in the cube. These 2 columns clearly highlight the quality and relevance of the approximation using $\mathit{Edge\text{-}Sky}(M)$. For most of datasets, the ratio is less than 5% (sometimes less than 1%), and of very small size compared to the approximation using closed patterns.

Table V focusses on the 3 datasets having the largest cubes: german, hypo and mushroom. For $|M|{=}k$, reported values are the average values over all $\binom{5}{k}$ possible skypattern cubes. Columns 5 and 6 confirm the results depicted at Table IV. Extra patterns always represent less than 1%.

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

Dataset	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{T}\rvert$	density	(1)	(2)	(3)	(4)	(5)	(6)
austral	55	690	0.272	6m04s	4m15s	6m00s	1m31s	2m18s	**36s**
cleve	43	303	0.325	1m53s	1m21s	28s	21s	18s	**10s**
cmc	28	1,474	0.357	26s	2m23s	31s	22s	**10s**	22s
crx	59	690	0.269	8m40s	5m37s	1m49s	1m13s	1m49s	**40s**
german	76	1,000	0.276	2h34m18s	53m29s	1h33m19s	14m03s	1h16m11s	**10m29s**
heart	38	270	0.368	1m46s	58s	38s	19s	29s	**15s**
horse	75	300	0.235	10m34s	3m32s	3m43s	58s	2m28s	**41s**
hypo	47	3,163	0.389	6h13m57s	51m46s	1h11m19s	44m41s	1h16m19s	**38m58s**
lymph	59	142	0.322	4m32s	49s	1m34s	31s	34s	**19s**
mushroom	119	8,124	0.193	1h53m39s	9h23m28s	55m23s	8h54m43s	**45m51s**	2h00m77s
tic-tac-toe	29	259	0.344	1m10s	2m48s	53s	41s	41s	**16s**
vehicle	58	846	0.327	34m01s	16m41s	5m45s	2m55s	3m32s	**1m42s**
wine	45	179	0.311	1m00s	31s	44s	13s	19s	**5s**
zoo	43	102	0.394	19s	8s	3s	3s	**2s**	**2s**

(1) Base-Line-Aetheris (3) MICMAC+Orion (5) Edge-Aetheris+Orion
(2) Base-Line-CP+SKY (4) CP+SKY+CUBE (6) CP+Edge-SKY+Orion

TABLE III: CPU-TIMES (UCI DATASETS).

Dataset	$nCube(M)$	$nEdge(M)$	$nClosed(M)$	(5)	(6)
austral	100,534.00	107,632.00	337,399.00	0.07	0.68
cleve	37,021.00	37,105.00	99,840.00	0.00	0.63
cmc	11,744.00	11,791.00	26,862.00	0.00	0.56
crx	102,690.00	104,377.00	409,615.00	0.02	0.75
german	1,138,533.00	1,149,315.00	4,715,852.00	0.01	0.76
heart	33,355.00	34,269.00	91,020.00	0.03	0.62
horse	119,446.00	119,587.00	297,627.00	0.00	0.60
hypo	671,929.00	677,051.00	1,605,518.00	0.01	0.58
lymph	45,799.00	54,880.00	161,447.00	0.17	0.66
mushroom	554,943.00	555,377.00	1,708,099.00	0.00	0.67
tic-tac-toe	15,020.00	15,310.00	49,822.00	0.02	0.69
vehicle	126,322.00	127,682.00	805,311.00	0.01	0.84
wine	13,835.00	13,941.00	47,846.00	0.01	0.71
zoo	4,375.00	4,509.00	16,697.00	0.03	0.73

(5) $1 - \frac{nCube(M)}{nEdge(M)}$ (6) $1 - \frac{nCube(M)}{nClosed(M)}$

TABLE IV: EFFECTIVENESS OF THE APPROXIMATION (UCI DATASETS).

$\lvert M\rvert$	$nCube(M)$	$nEdge(M)$	$nClosed(M)$	(5)	(6)
german					
2	315,280.00	315,646.00	3,942,910.00	0.00	0.92
3	556,458.00	558,325.00	4,242,910.00	0.00	0.87
4	835,367.00	840,539.00	4,500,560.00	0.01	0.81
5	1,138,533.00	1,149,315.00	4,715,852.00	0.01	0.76
hypo					
2	245,748.00	246,882.00	1,267,060.00	0.00	0.81
3	380,205.00	383,674.00	1,453,550.00	0.01	0.74
4	522,325.00	527,288.00	1,566,360.00	0.01	0.66
5	671,929.00	677,051.00	1,605,518.00	0.01	0.58
mushroom					
2	108,658.00	108,675.00	920,010.00	0.00	0.88
3	236,782.00	236,860.00	1,216,090.00	0.00	0.81
4	391,253.00	391,465.00	1,478,786.00	0.00	0.74
5	554,943.00	555,377.00	1,708,099.00	0.00	0.67

(5) $1 - \frac{nCube(M)}{nEdge(M)}$ (6) $1 - \frac{nCube(M)}{nClosed(M)}$

TABLE V: ZOOMING ON THE THREE SELECTED DATASETS.

VI. CONCLUSION

We have proposed an approximation based approach to compute skypattern cubes using soft skypatterns. Experiments show that our approach clearly outperforms the unique existing approach and enables to compute skypattern cubes of larger dimension. Navigation through the cube is a highly promising perspective, equivalence classes being able to give prominence the measures having the same skypattern set.

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.

REFERENCES

[1] S. Börzsönyi, D. Kossmann, and K. Stocker, “The skyline operator,” in *ICDE*, 2001.

[2] A. Soulet, C. Raïssi, M. Plantevit, and B. Crémilleux, “Mining dominant patterns in the sky,” in *ICDM*, 2011.

[3] W. Ugarte, P. Boizumault, S. Loudni, B. Crémilleux, and A. Lepailleur, “Mining (soft-) skypatterns using dynamic CSP,” in *CPAIOR*, 2014.

[4] C. Raïssi, J. Pei, and T. Kister, “Computing closed skycubes,” *PVLDB*, vol. 3, no. 1, 2010.

[5] W. Ugarte, P. Boizumault, S. Loudni, and B. Crémilleux, “Computing skypattern cubes,” in *ECAI*, 2014.

[6] J. Pei, A. W.-C. Fu, X. Lin, and H. Wang, “Computing compressed multidimensional skyline cubes efficiently,” in *ICDE*, 2007.

[7] H. Mannila and H. Toivonen, “Levelwise search and borders of theories in knowledge discovery,” *DMKD*, 1997.

[8] B. Négrevergne, A. Dries, T. Guns, and S. Nijssen, “Dominance programming for itemset mining,” in *ICDM*, 2013.

[9] J. Pei, Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, and Q. Zhang, “Towards multidimensional subspace skyline analysis,” *ACM ToDS*, 2006.

[10] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, “Efficient computation of the skyline cube,” in *VLDB*, 2005.

[11] J. Pei, W. Jin, M. Ester, and Y. Tao, “Catching the best views of skyline: A semantic approach based on decisive subspaces,” in *VLDB*, 2005.

[12] M. van Leeuwen and A. Ukkonen, “Discovering skylines of subgroup sets,” in *ECML/PKDD*, 2013.

[13] M. L. Yiu, E. Lo, and D. Yung, “Measuring the sky: On computing data cubes via skylining the measures,” *IEEE TKDE*, vol. 24, no. 3, 2012.

[14] G. Verfaillie and N. Jussien, “Constraint solving in uncertain and dynamic environments: A survey,” *Constraints*, 2005.

[15] A. Soulet and B. Crémilleux, “Adequate condensed representations of patterns,” *DMKD*, vol. 17, no. 1, 2008.

[16] K. Hansen, S. Mika, T. Schroeter, A. Sutter, A. ter Laak, T. Steger-Hartmann, N. Heinrich, and K. Müller, “Benchmark data set for in silico prediction of ames mutagenicity,” *Journal of Chemical Information and Modeling*, 2009.