ECAI 2014
T. Schaub et al. (Eds.)
© 2014 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-419-0-903

Computing Skypattern Cubes

Willy Ugarte[1] and **Patrice Boizumault**[1] and **Samir Loudni**[1] and **Bruno Crémilleux**[1]

Abstract. We introduce skypattern cubes and propose an efficient bottom-up approach to compute them. Our approach relies on derivation rules collecting skypatterns of a parent node from its child nodes without any dominance test. 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. Experiments show the effectiveness of our proposal.

1 Introduction

The notion of skyline queries [1] has been recently integrated into the pattern discovery paradigm to mine skyline patterns (henceforth called *skypatterns*) [10, 11]. 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 may prefer a pattern with a high frequency, large size and a high confidence. In this example, 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 to the skypatterns a global interest with semantics easily understood by the user.

In practice, users do not know the exact role of each measure which be used and it is difficult to beforehand select the most appropriate set of measures. 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 [9], users would like to have available 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.

More formally, given a set M of n measures, the 2^n-1 possible non-empty skypattern subsets should be precomputed to efficiently handle various queries of users. A baseline method to build the skypattern cube needs the computing of the skypatterns on every measure subset and incurs a prohibitive cost. Therefore the problem of efficient computing of the skypattern cube is the focus of this paper.

For computing the skypattern cube, we propose a bottom-up approach motivated by the following observations. First, we formally give two derivation rules providing an easy way to automatically infer a large proportion of the skypatterns of a *parent node* from the skypattern sets of its *child nodes* without any dominance test (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). For the new skypatterns of a parent node (i.e., skypatterns which are not skypatterns in its child nodes), we give an efficient technique based on dynamic CSP to mine them on the fly. We show that in practice the number of new skypatterns remains low. Second, we demonstrate how the bottom-up principle enables us to determine skypattern equivalence classes without any supplementary effort. This result has the advantage to provide a more concise cube, highlighting the measures giving the same skypattern set. Third, experiments conducted on real-life datasets show the practical effectiveness achieved by our formal results. To sum up, to the best of our knowledge, we designed the first method to build the skypattern cube without enumerating and mining all the possible skypatterns.

This paper is organized as follows. After introducing the background in Section 2, we present in Section 3 the formal properties to automatically infer skypatterns and build the concise representation of the cube. Section 4 describes our CSP method to mine the new skypatterns. We discuss related work in Section 5. Section 6 presents the experiments and we conclude in Section 7.

2 Skypattern Cube

2.1 Context and Definitions

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 $\mathcal{L}_{\mathcal{I}}$. Fig. 1a depicts a transactional dataset $\mathcal{T}$ where each transaction (or pattern) t_i is 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 is a product bought by the customer. An attribute (*price*) is associated to each product (see Fig. 1a).

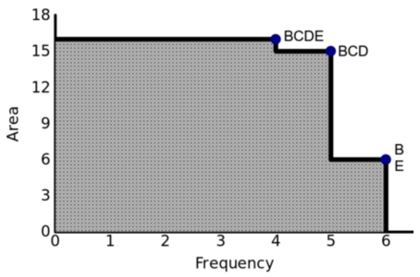
Constraint-based pattern mining aims at extracting all patterns x of $\mathcal{L}_{\mathcal{I}}$ satisfying a query $q(x)$ (conjunction of constraints) which is usually called *theory* [5]: $Th(q) = \{x \in \mathcal{L}_{\mathcal{I}} \mid q(x)\ is\ true\}$. A common example is the frequency measure leading 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:

[1] GREYC (CNRS UMR 6072), University of Caen Basse-Normandie, Boulevard du Maréchal Juin, 14032 CAEN cedex 5 - France.

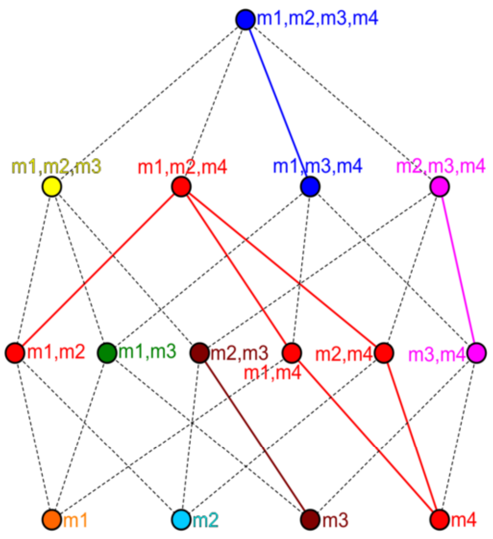
Trans.	Items					
t_1		B			E	F
t_2		B	C	D		
t_3	A				E	F
t_4	A	B	C	D	E	
t_5		B	C	D	E	
t_6		B	C	D	E	F
t_7	A	B	C	D	E	F

Item	A	B	C	D	E	F
Price	30	40	10	40	70	55

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



(b) Skypatterns for M={$freq, area$}.



(c) Lattice associated to M.

Subset of M	**Skypattern set**
$\{m_1, m_2, m_3, m_4\}$	{BCDE, BCD, BDE, EF, BE, E}
$\{m_1, m_2, m_3\}$	{BCDE, BCD, BE, E}
$\{m_1, m_2, m_4\}$	{E}
$\{m_1, m_3, m_4\}$	{BCDE, BCD, BDE, EF, BE, E}
$\{m_2, m_3, m_4\}$	{BCDE, BDE, EF, E}
$\{m_1, m_2\}$	{E}
$\{m_1, m_3\}$	{BCDE, BCD, B, E}
$\{m_1, m_4\}$	{E}
$\{m_2, m_3\}$	{BCDE}
$\{m_2, m_4\}$	{E}
$\{m_3, m_4\}$	{BCDE, BDE, EF, E}
$\{m_1\}$	{B, E}
$\{m_2\}$	{ABCDEF, ABCEF, ABDEF, ABEF, ABCDE, ABCE, ABDE, ABE, ACDEF, ACEF, ACDE, ACE, ADEF, ADE, AEF, AE, BCDEF, BCEF, CDEF, CEF, BCDE, BCE, CDE, CE, BDEF, DEF, BDE, DE, BEF, EF, BE, E}
$\{m_3\}$	{BCDE}
$\{m_4\}$	{E}

(d) Skypattern cube for M.

Figure 1: M={m_1: $freq, m_2$: max, m_3: $area, m_4$: $mean$}.

- $area(x) = freq(x) \times size(x)$.
- $min(x.att)$ (resp. $max(x.att)$) is the smallest (resp. highest) value of the set of item values of x for attribute att.
- $mean(x) = (min(x.att) + max(x.att))/2$.

Example 1 *For the dataset in Fig. 1a, freq(BC)=5, area(BC)=10 and mean(BCD.price)=25.*

A pattern x_i is closed w.r.t. a measure m iff $\forall\, x_j \supsetneq x_i, m(x_j) \neq m(x_i)$. 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.

2.2 Skypatterns

As stated above, skypatterns enable to express a user-preference point of view according to a dominance relation [10].

Definition 1 (Pareto Dominance) *Given a set of measures M, 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 2 (Skypattern and skypattern operator) *Given a set of measures M, 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: $Sky(M) = \{x_i \in \mathcal{L_I} \mid \not\exists x_j \in \mathcal{L_I}, x_j \succ_M x_i\}$*

Example 2 *From $\mathcal{T}$ and with M={$freq, area$}, BCD dominates BC as $freq(BCD)$=$freq(BC)$=5 and $area(BCD)$>$area(BC)$ (cf. Fig. 1a). Fig. 1b provides a graphical representation of $Sky(M) = \{BCDE, BCD, B, E\}$. The shaded area is called the forbidden area since it cannot contain any skypattern. The other part is called the dominance area.*

Let M be a set of measures. 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 3 (Incomparable Skypattern) *A pattern $x \in Sky(M)$ is incomparable w.r.t M iff $\forall x_i \in Sky(M)$ s.t. $x_i \neq x$, $x_i \prec\succ_M x$.*

Definition 4 (Indistinct Skypattern) *A pattern $x \in Sky(M)$ is indistinct w.r.t. M iff $\exists x_i \in Sky(M)$ s.t. $(x_i \neq x) \wedge (x_i =_M x)$.*

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. This remark will be precious for the derivation rule on indistinct skypatterns.

Definition 5 (Indistinct Skypattern Group (ISG)) *$S \subseteq Sky(M)$ is an indistinct skypattern group w.r.t. M, iff $|S| \geq 2$ and $\forall x_i, x_j \in S$, $(x_i =_M x_j)$ and $\forall x_i \in S$, $\forall x_j \in Sky(M) \setminus S$, $(x_i \prec\succ_M x_j)$.*

Example 3 *For M={$freq, area$}, $BCDE$ and BCD are incomparable. B and E are indistinct and belong to the same ISG.*

2.3 Skypattern Cube

Let M be a set of measures. We define the skypattern cube over M which consists of the $2^{|M|}-1$ skypattern sets $Sky(M_u)$ on all possible non-empty subset $M_u \subseteq M$.

Definition 6 (Skypattern Cube) *Let M be a set of measures. $SkyCube(M) = \{(M_u, Sky(M_u)) \mid M_u \subseteq M, M_u \neq \emptyset\}$.*

Example 4 *Consider the dataset in Fig. 1a. Fig. 1c depicts the lattice associated to M. Fig. 1d associates to each non-empty subset of M its skypattern set.*

3 Derivation Rules and Concise Representation

This section presents our *bottom-up* approach for computing the skypattern cube. The key idea is to collect the skypatterns of a parent node from the skypatterns of its child nodes. Then we compute the missing skypatterns of the node (i.e., skypatterns that are not skypatterns in its child nodes and thus not yet mined). We show how a concise representation of the cube is straightforwardly provided.

3.1 Derivation Rules

Theorems 1 states that all the incomparable skypatterns of a child node remain incomparable skypatterns in its parent nodes. Theorem 2 exhibits the indistinct skypatterns of a child node that remain skypatterns in its parent nodes. These two theorems define two derivation rules that enable to derive a subset of skypatterns of a parent node.

Theorem 1 (Incomparability Rule) *Let $M_u \subseteq M$. If x is an incomparable skypattern w.r.t. M_u then $\forall m \in M\backslash M_u$, $x \in Sky(M_u \cup \{m\})$. Moreover x is incomparable w.r.t. $M_u \cup \{m\}$.*

Proof (By Contradiction) Assume that x is an incomparable skypattern w.r.t. M_u, and $\exists m \in M\backslash M_u$ s.t. $x \notin Sky(M_u \cup \{m\})$. So, $\exists y{\neq}x \in \mathcal{L_I}$, $(y \succ_{M_u \cup \{m\}} x)$ i.e. (1) $\forall m_i \in M_u \cup \{m\}, m_i(y) \geq m_i(x)$ and (2) $\exists m_j \in M_u \cup \{m\}, m_j(y) > m_j(x)$. For (2), there are two cases:

1. $(m_j{=}m)$. As $x \in Sky(M_u)$, $\forall m_i \in M_u, m_i(x) \geq m_i(y)$. From (1), we deduce: $\forall m_i \in M_u, m_i(x) = m_i(y)$. So x is indistinct w.r.t. M_u. It contradicts that x is incomparable w.r.t. M_u.
2. $(m_j \in M_u)$. From (1), we have $\forall m_i \in M_u, m_i(y) \geq m_i(x)$. As, $m_j(y) > m_j(x)$, we deduce that $y \succ_{M_u} x$. It contradicts that x is a skypattern w.r.t. M_u. ■

Example 5 *Let $M_u{=}\{m_1, m_3\}$. $BCDE$ and BCD are incomparable w.r.t. M_u. Theorem 1 enables to deduce that $BCDE$ and BCD belong to $Sky(M_u \cup \{m_2\})$ and $Sky(M_u \cup \{m_4\})$.*

Theorem 2 (ISG Rule) *Let $M_u \subseteq M$ and S an ISG w.r.t. M_u. $\forall m \in M\backslash M_u$, each skypattern $x \in S$ s.t. $m(x) = \max_{x_i \in S}\{m(x_i)\}$ is a skypattern w.r.t. $M_u \cup \{m\}$.*

Proof (By Contradiction) Assume that there exists an ISG S w.r.t. M_u s.t. $\exists m{\in}M\backslash M_u$ s.t. $\exists x{\in}S$ s.t. $m(x){=}\max_{x_i \in S}\{m(x_i)\}$ and $x{\notin}Sky(M_u{\cup}\{m\})$. So, $\exists y{\neq}x \in \mathcal{L_I}$, $(y \succ_{M_u \cup \{m\}} x)$ i.e. (1) $\forall m_i \in M_u \cup \{m\}, m_i(y) \geq m_i(x)$ and (2) $\exists m_j \in M_u \cup \{m\}, m_j(y) > m_j(x)$. For (2), there are two cases:

1. $(m_j{=}m)$. As $x \in Sky(M_u)$, $\forall m_i \in M_u, m_i(x) \geq m_i(y)$. From (1), we deduce: $\forall m_i \in M_u, m_i(x) = m_i(y)$, i.e. $y \in S$. So, $m(y) \leq m(x)$ (as $m(x) = \max_{x_i \in S}\{m(x_i)\}$. It contradicts (2).
2. $(m_j \in M_u)$. From (1), we have $\forall m_i \in M_u, m_i(y) \geq m_i(x)$. As, $m_j(y) > m_j(x)$, we deduce that $y \succ_{M_u} x$. It contradicts that x is a skypattern w.r.t. M_u. ■

Example 6 *$S = \{B, E\}$ is an ISG w.r.t. $\{m_1\}$. Theorem 2 enables to deduce that:*

- *$E \in Sky(\{m_1, m_2\})$ since $m_2(E) = \max_{x_k \in S}\{m_2(x_k)\}$*
- *$E \in Sky(\{m_1, m_4\})$ since $m_4(E) = \max_{x_k \in S}\{m_4(x_k)\}$*
- *$B, E \in Sky(\{m_1, m_3\})$ since $m_3(B){=}m_3(E){=}\max_{x_k \in S}\{m_3(x_k)\}$*

Corollary 1 Let S an ISG w.r.t. M_u, and $m \in M\backslash M_u$. 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 $M_u \cup \{m\}$.

3.2 Computing a Skypattern Cube

The skypatterns of a node are computed in two steps. First, we collect all the skypatterns which can be derived from its child nodes. Then the missing skypatterns (i.e., non-derivable skypatterns) are computed. We start by defining the *derivable skypatterns*.

Let $M_u \subseteq M$ and $m \in M\backslash M_u$. We define $inc(M_u)$ the set of incomparable skypatterns w.r.t M_u and $ind(M_u, m)$ the set of maximal indistinct skypatterns w.r.t a measure m:

- $inc(M_u) = \{x \in Sky(M_u) \mid x\ is\ incomparable\ w.r.t.\ M_u\}$
- $ind(M_u, m) = \bigcup_{ISG\ S \subseteq Sky(M_u)} \{x \in S \quad \mid m(x) = \max_{x_k \in S}\{m(x_k)\}\}$

$derived(M_u)$ is the set of skypatterns of the node associated to M_u which can be derived from the skypatterns of its child nodes:

- $derived(M_u) = \bigcup_{m \in M_u} (inc(M_u\backslash\{m\}) \cup ind(M_u\backslash\{m\}, m))$.

First, it is obvious that: $derived(M_u) \subseteq Sky(M_u)$ (see Theorem 1 and 2). Experiments show that a large proportion of skypatterns are obtained by this way. Moreover, if a skypattern x in a parent node is also a skypattern in at least one of its child nodes, then x will be necessary collected by one of these rules. It is expressed by:
$derived(M_u) = (\bigcup_{m \in M_u} Sky(M_u \setminus \{m\})) \cap Sky(M_u)$.

This property shows the power of our derivation rules. (The proof is immediate since, for each node, incomparable and indistinct skypatterns constitute a partition.) However, $derived(M_u)$ can be strictly included in $Sky(M_u)$, i.e., some skypatterns are missing. It happens when a skypattern of a node is not a skypattern in any of its child nodes as illustrated by the following example.

Example 7 *As $BCDE$ is incomparable w.r.t. $\{m_3\}$, $BCDE \in Sky(\{m_1, m_3\})$. As B and E constitute an ISG w.r.t. $\{m_1\}$ and $m_3(B){=}m_3(E)$, then $B, E \in Sky(\{m_1, m_3\})$. But, the derivation rules cannot deduce that $BCD \in Sky(\{m_1, m_3\})$.*

We compute on the fly the non-derivable skypatterns thanks to a dynamic CSP method described in Section 4. Moreover, we can go further by detecting a priori that $derived(M_u) = Sky(M_u)$ for some M_u and thus avoiding useless computation. Theorem 3 states a sufficient condition ensuring that $derived(M_u) = Sky(M_u)$. Experiments show that this condition is effective in practice.

Theorem 3 (Non-Computing Sufficient Condition) *Let $M_u{\subseteq}M$. If $\exists m \in M_u$ s.t. $\min_{x \in derived(M_u)}\{m(x)\} = \max_{x \in derived(M_u)}\{m(x)\}$ then $Sky(M_u) = derived(M_u)$.*

Proof (By Contradiction) Let $m \in M_u$ a measure s.t. $\min_{x \in derived(M_u)}\{m(x)\}{=}\max_{x \in derived(M_u)}\{m(x)\}$
Assume that $\exists p \in Sky(M_u) \setminus derived(M_u)$, so $m(p){=}\min_{x \in derived(M_u)}\{m(x)\}{=}\max_{x \in derived(M_u)}\{m(x)\}$. Henceforth $m(p){=}\max_{x \in \mathcal{L_I}}\{m(x)\}$. Thus, $p \in Sky(\{m\})$, and:

(i) either p is incomparable w.r.t. $\{m\}$,
(ii) or p is indistinct w.r.t $\{m\}$ with maximal value for m.

From (i) and (ii), $p \in derived(M_u)$ leading to a contradiction. ■

Finally, Algorithm 1 gives the pseudo-code of our bottom-up approach. It starts by computing $Sky(\{m\})$ for every $m \in M$ and then follows a level-wise strategy: from the lower level, each level of the lattice is constructed by applying the derivation rules and, if needed, the computing of the non derivable skypatterns (function *complete*).

3.3 Concise Representation of a Cube

Different subsets of measures may lead to a same set of skypatterns. This observation can be used to provide a concise representation of the cube without loss of information. We define an equivalence relation over subsets of measures having the same skypattern set:

```
Algorithm 1: Bottom-up approach for computing the cube
  Input: $M$: a set of measures, $\mathcal{T}$: a dataset.
  Output: The skypattern cube of dataset $\mathcal{T}$ w.r.t. $M$.
1 cube ← $\emptyset$;
2 foreach $m \in M$ do
3     cube ← cube $\cup \{(\{m\}, Sky(\{m\}))\}$;
4 for $i \leftarrow 2$ to $|M|$ do
5     foreach $M_u \subset M$ s.t. $|M_u| = i$ do
6         cube ← cube $\cup \{(M_u, complete(derived(M_u)))\}$;
7 return cube
```

Definition 7 (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 8 *Equivalence classes on our running example are illustrated in Figure 1c. There are 8 classes: 4 have a cardinality of 1, 3 have a cardinality of 2 and 1 has a cardinality of 5.*

Theorem 4 indicates if a new node built by the addition of a measure to a subset of measures M_u belongs or not to the equivalence class of M_u. It means that equivalence classes can be easily determined thanks to the bottom-up construction of the skypattern cube. In other words, when our approach is running to extract the skypatterns of the cube, it can also provide a concise representation of the cube without supplementary work.

Theorem 4 (Equivalence class) *Let $M_u \subseteq M$ and $m \in M \backslash M_u$. $Sky(M_u \cup \{m\}) = Sky(M_u)$ iff (1) all indistinct skypatterns w.r.t. M_u are indistinct skypatterns w.r.t. $M_u \cup \{m\}$ and (2) $Sky(M_u \cup \{m\}) = derived(M_u \cup \{m\})$.*

Proof (Double inclusion) All incomparable w.r.t. M_u are incomparable w.r.t. $M_u \cup \{m\}$ (Theorem 1). All indistinct w.r.t. M_u are indistinct w.r.t. $M_u \cup \{m\}$ according to (1). Since incomparable w.r.t. M_u and indistinct w.r.t. M_u form a partition of $Sky(M_u)$, $Sky(M_u) \subset Sky(M_u \cup \{m\})$.
According to (2), $Sky(M_u \cup \{m\}) = derived(M_u \cup \{m\})$. As derived skypatterns w.r.t. $M_u \cup \{m\}$ can only come from $Sky(M_u)$, $Sky(M_u \cup \{m\}) \subset Sky(M_u)$. ■

4 Mining non-derivable Skypatterns using DCSP

This section describes how the non-derivable skypatterns can be mined using Dynamic CSP [12]. The main idea of our approach [11], taking benefit from cross-fertilization between CSP and data mining [3, 4], is to improve the mining step during the process thanks to constraints dynamically posted and stemming from the current set of the candidate skypatterns. The process stops when the forbidden area cannot be enlarged. Finally, the completeness of our approach is insured by the completeness of the CP solver.

4.1 Mining Skypatterns

A Constraint Satisfaction Problem (CSP) $P=(\mathcal{X}, \mathcal{D}, \mathcal{C})$ is defined by a set of variables $\mathcal{X}$, a domain $\mathcal{D}$, which maps every variable $x_i \in \mathcal{X}$ to a finite set of values $D(x_i)$, and a set of constraints $\mathcal{C}$.

A Dynamic CSP [12] is a sequence $P_1, P_2, ..., P_n$ of CSP, each one resulting from some changes in the definition of the previous one. These changes may affect every component in the problem definition: variables, domains and constraints. *For our approach, changes are only performed by adding new constraints.* Solving such dynamic CSP involves solving a single CSP with additional constraints posted during search. Each time a new solution is found, new constraints are imposed. Such constraints will survive backtracking and state that next solutions should verify both the current set of constraints and the added ones.

Constraints on the dominance relation are dynamically posted during the mining process. Variable x will denote the (unknown) skypattern we are looking for. Changes are only performed by adding new constraints. So, we consider the sequence $P_1, P_2, ..., P_n$ of CSP where M is a set of measures, each $P_i = (\{x\}, \mathcal{L}_{\mathcal{I}}, q_i(x))$ and:

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

First, the constraint $closed_M(x)$ states that x must be a closed pattern w.r.t M, it allows to reduce the number of redundant patterns (see Section 2.1). Then, the constraint $\phi_i(x) \equiv \neg(s_i \succ_M x)$ states that the next solution (which is searched) 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 $s_1, s_2, \ldots, s_i$.

Each time the first solution s_i to query $q_i(x)$ is found, a new constraint $\phi_i(x)$ is dynamically posted, leading to reduce the search space. This process stops when the forbidden area cannot be further extended (i.e. there exists n s.t. query $q_{n+1}(x)$ has no solution). For skypatterns, $\phi_i(x)$ states that $\neg(s_i \succ_M x)$:
$\phi_i(x) \equiv \left(\bigvee_{m \in M} m(s_i) < m(x)\right) \vee \left(\bigwedge_{m \in M} m(s_i) = m(x)\right)$

But, the n extracted patterns $s_1, s_2, \ldots, s_n$ are not necessarily all skypatterns. Some of them can only be "intermediate" patterns simply used to enlarge the forbidden 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-steps approach:

1. Compute the set $S = \{s_1, s_2, \ldots, s_n\}$ of candidates using Dynamic CSP.
2. Remove 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 (see [11]).

4.2 Mining the non-derivable Skypatterns

In order to find the non-derivable skypatterns, we proceed in the same way as Section 4.1, by stating that any non-derivable skypattern could not de dominated by any derived skypattern.

Let $M_u \subseteq M$ and $derived(M_u)$ the subset of $Sky(M_u)$ obtained using the two derivation rules. Consider the sequence $P_1, P_2, ..., P_n$ of CSP where each $P_i = (\{x\}, \mathcal{L}_{\mathcal{I}}, q_i(x))$ and:

- $q_1(x) = closed_{M_u}(x) \wedge \Psi_{M_u}(x)$
- $q_{i+1}(x) = q_i(x) \wedge \neg(s_i \succ_{M_u} x)$ where s_i is the first solution to query $q_i(x)$
- $\Psi_{M_u}(x)$ states that x cannot be dominated w.r.t. M_u by any derived skypattern:

$$\Psi_{M_u}(x) = \bigwedge_{x_i \in derived(M_u)} \neg(x_i \succ_{M_u} x)$$

Example 9 *Consider example 7. For $M_u = \{m_1, m_3\}$, $derived(M_u) = \{B, E, BCDE\}$. So $\Psi_{M_u}(x) = \neg(B \succ_{M_u} x) \wedge \neg(E \succ_{M_u} x) \wedge \neg(BCDE \succ_{M_u} x)$. The associated Dynamic CSP has a unique solution: $x = BCD$.*

5 Related Work

Mining skypatterns is different from mining skylines [1]. Skyline queries focus on the extraction of tuples of the dataset and assume that all skylines are in the dataset. The skypattern mining task consists in 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: $O(2^{|\mathcal{I}|})$ instead of $O(|\mathcal{T}|)$ for skylines.

Two methods have been designed for mining skypatterns. `Aetheris` [10] 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 2) is applied. `CP+SKY` [11] mines skypatterns using Dynamic CSP (see Section 4.1). Both methods have the same efficiency, but `CP+SKY` also allows to mine soft skypatterns [11].

Computing the skyline cube efficiently. [7, 8, 13] proposed several strategies to share skyline computation in different subspaces, but they have to cope with the problem of enumerating skylines over all possible subspaces. [6] proposed *Stellar*, which computes seed skylines groups in the full space, then extend them to build the final set of skyline groups and thus avoiding the computation of skylines in all the subspaces. But *Stellar* does not take profit from any parent-child relationships in the lattice. [9] is able to decrease the number of domination tests by reducing the number of measure subspaces that needs to be searched. However, its complex strategy makes impossible the success of the full use of the parent-child relationships. Moreover, all of these techniques address skylines.

6 Experimental Evaluation

6.1 Skypattern Cubes for Mutagenicity Dataset

In this section, we report an experimental evaluation on a real-life dataset of large size extracted from mutagenicity data [2] (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 of the chemicals.

$\|M\|$	(2)	$\frac{(2)}{2^{\|M\|}-1}$	(4)	(5)	$\frac{(4)}{(5)}$	(7)	Speed-up
1	1	1.00	338	338	1.00	1m:33s	1.00
2	2	0.87	680	753	0.90	14m:39s	1.19
3	5	0.75	1,036	1,280	0.80	28m:12s	1.70
4	9	0.64	1,421	1,983	0.71	48m:43s	2.40
5	16	0.53	1,865	2,982	0.62	1h:19m:30s	3.37
6	28	0.44	2,424	4,526	0.53	2h:04m:45s	4.73
7	45	0.36	3,200	7,146	0.45	3h:09m:35s	6.69
8	73	0.29	4,386	12,015	0.36	4h:40m:03s	9.57
9	117	0.23	6,327	21,773	0.23	6h:43m:07s	13.93
10	213	0.20	9,619	42,386	0.23	9h:26m:42s	20.62
11	401	0.20	15,261	87,374	0.17	12h:59m:36s	30.97

(2) # of equivalence classes (5) $\sum_{M_u \subseteq M} | Sky(M_u) |$

(4) # of skypatterns for the concise representation (7) CPU-Time for CP+SKY+CUBE

Table 1: Space analysis (left part) and CPU-time analysis (right part).

Experimental protocol. The implementation of `CP+SKY+CUBE` (bottom-up approach proposed in this paper) was carried out in

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

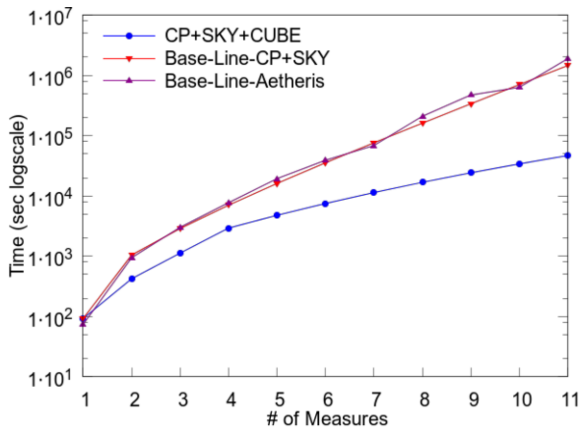


Figure 2: Comparing CPU-times for the 3 methods.

`Gecode` by extending the CP-based pattern extractor developed by [4]. All experiments were conducted on a computer running Linux with a core i3 processor at 2.13 GHz.

(a) CPU-time analysis. We compare `CP+SKY+CUBE` with two other methods for computing a skypattern cube:

1. `Base-Line-Aetheris` applies `Aetheris` to the $2^{|M|}-1$ non empty subsets of M.
2. `Base-Line-CP+SKY` applies `CP+SKY` to the $2^{|M|}-1$ non empty subsets of M.

For the base-line methods, the CPU-time is the sum of CPU-times required for each non-empty subset of M.

Fig. 2 compares the performance of the three methods according to the number of measures $|M|$. The scale is logarithmic. 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. `Base-Line-Aetheris` and `Base-Line-CP+SKY` have a similar behavior since `Aetheris` and `CP+SKY` are equally effective (see Section 5). `CP+SKY+CUBE` clearly outperforms the two base-line methods. For a small number of measures $(2 \leq |M| \leq 4)$, the speed-up is 1.75 (see column 8, Table 1). For $|M|=8$, there is an order of magnitude (speed-up value 9.57). For $|M|=11$, `CP+SKY+CUBE` requires about 13 hours to compute the skypattern cube, while the two baseline methods spent about 403 hours (speed-up value 31). Finally, if $|M|$ is increased by one (a new measure is added), the number of subsets to consider is doubled but we can see that our speed-up is multiplied by about 1.5 (see column 8, Table 1).

(b) Space analysis. Column 1 (Table 1) corresponds to the number of measures. Column 2 indicates the number of equivalence classes. Column 3 denotes the ratio between the number of equivalent classes and the total number of subsets of measure. Column 4 (resp. 5) reports the total number of skypatterns for the concise (resp. usual) representation (see Section 3.3) and Column 6 gives their ratio. For $|M|=k$, reported values in columns (2), (4) and (5) represent the averages over all $\binom{11}{k}$ possible skypattern cubes. Our concise representation of the skypattern cube provides a substantial summarization and compression of skypattern sets. For instance, for $|M|=11$, there are 401 classes and a total number of $15,261$ skypatterns for the concise representation. For the usual representation, there are $2,047$ subsets of measure and and a total number of $87,374$ skypatterns. This leads to a substantial gain greater than 80%.

(c) Effectiveness of our derivation rules. In order to evaluate the effectiveness of the two derivation rules (cf. Section 3.1), we measured the percentage of derived skypatterns (vs the total number of

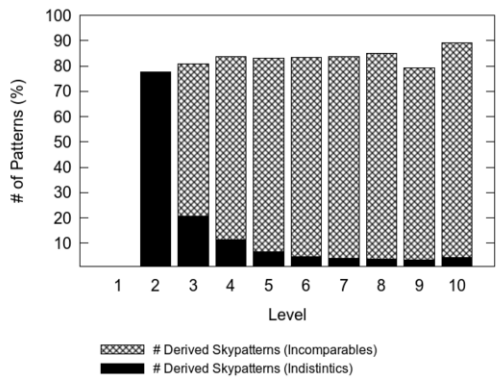


Figure 3: Effectiveness of our derivation rules.

skypatterns) at each level of a cube. Reported values in Fig. 3 are average values over all 11 possible cubes of 10 measures. For each level i $(2 \leq i \leq 10)$, the proportion of incomparable and indistinct skypatterns is also depicted.

Our derivation rules are very efficient since they are able to deduce about $80 - 90\%$ of the skypatterns, except for the first levels. Moreover, when the number of measures increases, the number of indistinct skypatterns decreases (in percentage), while the number of incomparable skypatterns increases. Indeed, incomparable skypatterns of child nodes remain incomparable for a parent node (see Theorem 1) whereas indistinct skypatterns may become any kind of skypatterns or dominated patterns (see Corollary 1).

(d) Effectiveness of our sufficient condition. Theorem 3 gives our sufficient condition stating if, for a subset of measures M_u, $derived(M_u) = Sky(M_u)$ without requiring any Dynamic CSP. To asses the effectiveness of this condition, we measured the percentage of success at each level of a cube. Reported values in Table 2 are average values over all 11 possible cubes of 10 measures. Line 1 provides the level in the lattice. Line 2 indicates the number of nodes where our condition applies, while Line 3 reports the number of nodes where our condition should apply. Line 4 depicts their ratio (percentage of success). The more the number of measures increases, the more our sufficient condition becomes effective. From level 5 to level 10, the percentage of success increases from 73% to 100%.

Level	2	3	4	5	6	7	8	9	10
# of nodes where Th 3 applies	6.55	54.55	127.91	183.27	171.82	106.91	42.55	9.82	1.00
# of nodes where Th 3 should apply	29.45	102.55	194.73	252.00	210.00	120.00	45.00	10.00	1.00
Succ. ratio: (2)/(3)	0.22	0.53	0.66	0.73	0.82	0.89	0.95	0.98	1.00

Table 2: Effectiveness of our sufficient condition (Theorem 3).

6.2 Skypattern Cubes for UCI Datasets

Experiments were carried out on 14 various datasets from UCI[3] benchmarks. We considered 5 measures $M = \{freq, max, area, mean, growth\text{-}rate\}$. Measures using numeric values, like $mean$, were applied on attribute values that were randomly generated within the range $[0..1]$. Table 3 summarizes the results we obtained.

(a) CPU-time analysis. Columns 2-4 compare the CPU-times of the three methods. `CP+SKY+CUBE` clearly dominates the two base-line methods. On half of the datasets, the speed-up is at least 10.43 (see column 5).

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

	CPU-Time			Speed-Up		
Dataset	(2)	(3)	(4)	$\frac{(2)}{(4)}$	$\frac{(3)}{(4)}$	(7)
austral	6m04s	4m15s	1m31s	3.98	2.79	0.82
cleve	1m53s	1m21s	21s	5.27	3.76	0.97
cmc	26s	2m23s	22s	1.20	6.41	0.90
crx	8m40s	5m37s	1m13s	7.12	4.61	0.89
german	2h34m18s	53m29s	14m03s	10.98	3.80	0.88
heart	1m46s	58s	19s	5.49	3.01	0.86
hepatic	6m12s	58s	19s	18.91	2.97	0.71
horse	10m34s	3m32s	58s	10.93	3.67	0.82
hypo	6h13m57s	51m46s	4m41s	79.75	11.04	0.79
lymph	4m32s	49s	11s	23.87	4.38	0.65
tic-tac-toe	1m10s	2m48s	41s	1.68	4.03	0.84
vehicle	34m01s	16m41s	2m55s	11.64	5.71	0.66
wine	1m00s	31s	13s	4.63	2.43	0.94
zoo	19s	8s	1s	10.43	4.82	0.87

(2) Base-line-Aetheris (4) CP+SKY+CUBE
(3) Base-line-CP+SKY (7) Succ. Ratio: $\frac{|derived(M_u)|}{|Sky(M_u)|}$

Table 3: Results on UCI datasets with $|M|$=5.

(b) Effectiveness of our derivation rules. Column 7 reports, for each dataset, the percentage of derived skypatterns. Reported values are the average values over levels i $(2 \leq i \leq 5)$ of the cube. Our derivation rules are able to deduce about $79 - 97\%$ of the skypatterns, except for two datasets (lymph and vehicle).

7 Conclusion

We have designed an efficient bottom-up method to compute skypattern cubes. Our derivation rules are able to collect a large part of the skypatterns of a parent node. Non-derivable skypatterns are computed on the fly thanks to Dynamic CSP. The bottom-up strategy makes easy the building of a concise representation of the cube according to skypattern equivalence classes. Experiments show the effectiveness of our proposal. Navigation through the cube is a highly promising perspective.

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*, pp. 421–430, (2001).
[2] K. Hansen et al., 'Benchmark data set for in silico prediction of ames mutagenicity', *J. of Chem. Inf. and Model.*, **49**(9), 2077–2081, (2009).
[3] T. Guns, S. Nijssen, and L. De Raedt, 'Itemset mining: A constraint programming perspective', *Artif. Intell.*, **175**(12-13), 1951–1983, (2011).
[4] M. Khiari, P. Boizumault, and B. Crémilleux, 'Constraint programming for mining n-ary patterns', in *CP*, LNCS 6308, pp. 552–567, (2010).
[5] H. Mannila and H. Toivonen, 'Levelwise search and borders of theories in knowledge discovery', *DAMI*, **1**(3), 241–258, (1997).
[6] J. Pei, A. W.-C. Fu, X. Lin, and H. Wang, 'Computing compressed multidimensional skyline cubes efficiently', in *ICDE*, pp. 96–105, (2007).
[7] J. Pei, W. Jin, M. Ester, and Y. Tao, 'Catching the best views of skyline: A semantic approach based on decisive subspaces', in *VLDB*, pp. 253–264, (2005).
[8] 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 Trans. Database Syst.*, **31**(4), 1335–1381, (2006).
[9] C. Raïssi, J. Pei, and T. Kister, 'Computing closed skycubes', *PVLDB*, **3**(1), 838–847, (2010).
[10] A. Soulet, C. Raïssi, M. Plantevit, and B. Crémilleux, 'Mining dominant patterns in the sky', in *ICDM*, pp. 655–664, (2011).
[11] W. Ugarte, P. Boizumault, S. Loudni, B. Crémilleux, and A. Lepailleur, 'Mining (soft-) skypatterns using dynamic CSP', in *CPAIOR*, LNCS 8451, pp. 71–87, (2014).
[12] G. Verfaillie and N. Jussien, 'Constraint solving in uncertain and dynamic environments: A survey', *Constraints*, **10**(3), 253–281, (2005).
[13] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, 'Efficient computation of the skyline cube', in *VLDB*, pp. 241–252, (2005).