Mining (Soft-) Skypatterns using Dynamic CSP

Willy Ugarte Rojas, Patrice Boizumault, Samir Loudni, Bruno Crémilleux, Alban Lepailleur

Mining (Soft-) Skypatterns using Constraint Programming

W. Ugarte[1], P. Boizumault[1], S. Loudni[1], B. Crémilleux[1], and A. Lepailleur[2]

[1] GREYC (CNRS UMR 6072) – University of Caen
Campus II Côte de Nacre, 14000 Caen - France
[2] CERMN (UPRES EA 4258 - FR CNRS 3038 INC3M) – University of Caen
Boulevard Becquerel, 14032 Caen Cedex - France

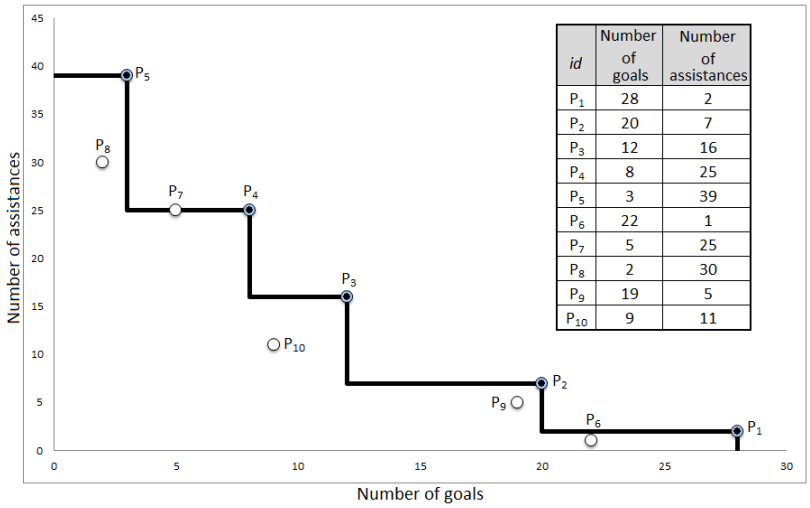
Abstract. Within the pattern mining area, skypatterns enable to express a user-preference point of view according to a dominance relation. In this paper, we deal with the introduction of softness in the skypattern mining problem. First, we show how softness can provide convenient patterns that would be missed otherwise. Then, thanks to Constraint Programming, we propose a generic and efficient method to mine skypatterns as well as soft ones. Finally, we show the relevance and the effectiveness of our approach through a case study in chemoinformatics.

1 Introduction

Discovering useful patterns from data is an important tool for data analysis and has been used in a wide range of applications. Many approaches have promoted the use of constraints to focus on the most promising knowledge according to a potential interest given by the final user. As the process usually produces a large number of patterns, a large effort is made to a better understanding of the fragmented information conveyed by the patterns and to produce *pattern sets* i.e. sets of patterns satisfying properties on the whole set of patterns [5].

Skyline queries [3] enable to express a user-preference point of view according to a *dominance* relation. In a multidimensional space where a preference is defined for each dimension, a point p_i *dominates* another point p_j if p_i is better (i.e., more preferred) than p_j in at least one dimension, and p_i is not worse than p_j on every other dimension. However, while this notion of skylines has been extensively developed and researched for database applications, it has remained unused until recently for data mining purposes. [17] proposes a technique to extract skyline graphs that maximize two measures (the number of vertices and the edge connectivity). The notion of skyline queries has been recently integrated into the constraint-based pattern discovery paradigm to mine skyline patterns (henceforth called *skypatterns*) [19]. As an example, a user may prefer a pattern with a high frequency, large length and a high confidence. In this case, we say that 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 interesting for a twofold reason: they do not require any threshold on the measures and the notion of dominance provides a global interest with semantics easily understood by the user.

Nevertheless, skypatterns queries, like other kinds of queries, suffer from the stringent aspect of the constraint-based framework. Indeed, a pattern satisfies or does not



id	Number of goals	Number of assistances
P_1	28	2
P_2	20	7
P_3	12	16
P_4	8	25
P_5	3	39
P_6	22	1
P_7	5	25
P_8	2	30
P_9	19	5
P_{10}	9	11

Fig. 1: A skyline example.

satisfy the constraints. But, what about patterns that slightly miss a constraint? The following example shows the interest of introducing softness. This example addresses *skylines in databases* because it is easier to illustrate the key points of introducing softness and to give rise the skypattern problem. Skypatterns and soft-skypatterns are formally introduced in the following sections. There are very few works such as [2, 21] which introduce softness into the mining process.

Consider a coach of a football team who looks for players for the next season (see Fig. 1). Every player is depicted according to the number of goals he scored and the number of assistances he performed during the last season. A point (here, a player) p_i *dominates* another point p_j if p_i is better than p_j in at least one dimension, and p_i is not worse than p_j on every other dimension. A skyline point is a point which is not dominated by any other point. The skyline set (or skyline for short) consists of players p_1, p_2, p_3, p_4 and p_5. Indeed, players p_6, p_7, p_8, p_9 and p_{10} are dominated by at least one other player, thus they cannot be part of the skyline. Nevertheless, the coach could be interested in non-skyline players if he looks for:

- players in a forward position: the coach gives the priority to the number of scored goals. The players p_1 (skyline), p_2 (skyline) are still interesting and p_6 (non-skyline) and p_9 (non-skyline) become interesting.
- players in an attacking midfielder position: the coach gives the priority to the number of performed assistances. The players p_4 (skyline) and p_5 (skyline) are still interesting and p_7 (non-skyline) and p_8 (non-skyline) become interesting.
- multipurpose players: the coach gives the priority to the trade-off between the number of scored goals and the number of performed assistances. The players p_3 (skyline) and p_4 (skyline) are still promising and p_{10} (non-skyline) becomes promising.

Moreover, skyline players are very sought and expensive: they might be signed by another team or their salaries could be out of budget. So, non-skyline players, that are close to skyline players, can be of great interest for the coach. Such promising players can be discovered by slightly relaxing the dominance relation.

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

Table 1: Transactional dataset $\mathscr{T}$.

The contributions of this paper are the following. First, we introduce the notion of soft skypattern. Second, we propose a flexible and efficient approach to mine skypatterns as well as soft ones thanks to the Dynamic CSP (Constraint Satisfaction Problems) framework [22]. Our proposition benefits from the recent progress on cross-fertilization between data mining and Constraint Programming (CP) [4, 9, 7]. The common point of all these methods is to model in a declarative way pattern mining as CSP, whose resolution provides the complete set of solutions satisfying all the constraints. We show how the (soft-)skypatterns mining problem can be modeled and solved using dynamic CSPs. A major advantage of the method is to improve the mining step during the process thanks to constraints dynamically posted and stemming from the current set of candidate skypatterns. Moreover, the declarative side of the CP framework leads to a unified framework handling softness in the skypattern problem. Finally, the relevance and the effectiveness of our approach is highlighted through a case study in chemoinformatics for discovering toxicophores.

This paper is organized as follows. Section 2 presents the context and defines skypatterns. Section 3 introduces soft skypatterns. Section 4 presents our flexible and efficient CP approach to mine skypatterns as well as soft ones. We review some related work in Section 5. Finally, Section 6 reports in depth a case study in chemoinformatics by performing both a performance and a qualitative analysis.

2 The skypattern mining problem

2.1 Context and definitions

Let $\mathscr{I}$ be a set of distinct literals called *items*. An itemset (or pattern) is a non-null subset of $\mathscr{I}$. The language of itemsets corresponds to $\mathscr{L}_{\mathscr{I}} = 2^{\mathscr{I}} \setminus \emptyset$. A transactional dataset $\mathscr{T}$ is a multiset of patterns of $\mathscr{L}_{\mathscr{I}}$. Each pattern (or transaction) is a database entry. Table 1 (left side) presents a transactional dataset $\mathscr{T}$ 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 Table 1, right side).

Constraint-based pattern mining aims at extracting all patterns x of $\mathscr{L}_{\mathscr{I}}$ satisfying a query $q(x)$ (conjunction of constraints) which is usually called *theory* [12]: $Th(q) = \{x \in \mathscr{L}_{\mathscr{I}} \mid q(x) \text{ is } true\}$. A common example is the frequency measure leading to the minimal frequency constraint. The latter provides patterns x having a number of occurrences in the dataset exceeding a given minimal threshold min_{fr}: $freq(x) \geq min_{fr}$. There are other usual measures for a pattern x:

- $size(x)$ is the number of items that x contains.

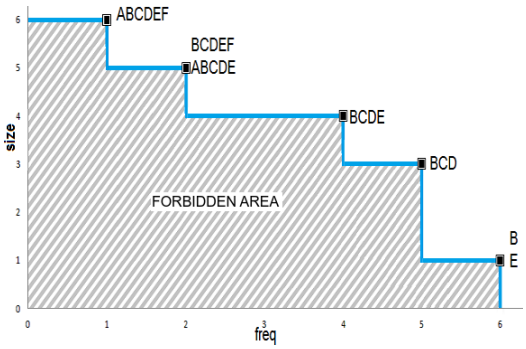


Fig. 2: Skypatterns extracted from the example in Table 1.

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

Considering the dataset described in Table 1, we have: $freq(BC)$=5, $size(BC)$=2 and $area(BC)$=10. Moreover, $average(BCD.price)$=30 and $mean(BCD.price)$=25.

In many applications, it is highly appropriated to look for contrasts between subsets of transactions, such as toxic and non toxic molecules in chemoinformatics (see Section 6). The growth rate is a well-used contrast measure [14].

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

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

Moreover, the user is often interested in discovering richer patterns satisfying properties involving several local patterns. These patterns define pattern sets [5] or n-ary patterns [9]. The approach presented in this paper is able to deal with such patterns.

2.2 Skypatterns

Skypatterns have been recently introduced by [19]. Such patterns enable to express a user-preference point of view according to a dominance relation. Given a set of patterns, the skypattern set contains the patterns that are not dominated by any other pattern.

Given a set of measures M, if a pattern x_j is dominated by another pattern x_i according to all measures of M, x_j is considered as irrelevant. This idea is at the core of the notion of skypattern.

Definition 2 (Dominance). *Given a set of measures M, a pattern x_i dominates another pattern x_j with respect to M (denoted by $x_i \succ_M x_j$), iff $\forall m \in M, m(x_i) \geq m(x_j)$ and $\exists m \in M, m(x_i) > m(x_j)$.*

Consider the example in Table 1 with M=$\{freq, area\}$. Pattern BCD dominates pattern BC because $freq(BCD)$=$freq(BC)$=5 and $area(BCD)$>$area(BC)$. For M=$\{freq, size, average\}$, pattern BDE dominates pattern BCE because $freq(BDE)$=$freq(BCE)$=4, $size(BDE)$=$size(BCE)$=3 and $average(BDE.price)$>$average(BCE.price)$.

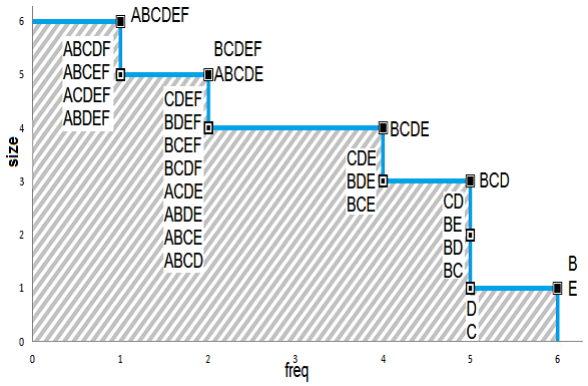


Fig. 3: Edge-skypatterns extracted from the example in Table 1.

Definition 3 (Skypattern operator). *Given a pattern set $P \subseteq \mathscr{L}_{\mathscr{I}}$ and a set of measures M, a skypattern of P with respect to M is a pattern not dominated in P with respect to M. The skypattern operator Sky(P,M) returns all the skypatterns of P with respect to M:* $Sky(P,M) = \{x_i \in P \mid \nexists x_j \in P, x_j \succ_M x_i\}$.

The skypattern mining problem is thus to evaluate the query $Sky(\mathscr{L}_{\mathscr{I}}, M)$. For instance, from the data set in Table 1 and with $M=\{freq, size\}$, $Sky(\mathscr{L}_{\mathscr{I}}, M) = \{ABCDEF, BCDEF, ABCDE, BCDE, BCD, B, E\}$ (see Figure 2). The shaded area is called the *forbidden area*, as it cannot contain any skypattern. The other part is called the *dominance area*. The edge of the dominance area (bold line) marks the boundary between these two zones.

The skypattern mining problem is challenging because of its NP-Completeness. There are $O(2^{|\mathscr{I}|})$ candidate patterns and a naive enumeration would lead to compute $O(2^{|\mathscr{I}|} \times |M|)$ measure values. [19] have proposed an efficient approach taking benefit of theoretical relationships between pattern condensed representations and skypatterns and making the process feasible when the pattern condensed representation can be extracted. Nevertheless, this method can only use a crisp dominance relation.

3 The soft skypattern mining problem

This section presents the introduction of softness in the skypattern mining problem. The skypatterns suffer from the stringent aspect of the constraint-based framework. In order to introduce softness in this context, we propose two kinds of soft skypatterns: the *edge-skypatterns* that belongs to the edge of the dominance area (see Section 3.1) and the *δ-skypatterns* that are close to this edge (see Section 3.2).

The key idea is to strengthen the dominance relation in order to soften the notion of non dominated patterns. The goal is to capture valuable skypatterns occurring in the forbidden area.

3.1 Edge-skypatterns

Similarly to skypatterns, edge-skypatterns are defined according to a dominance relation and a *Sky* operator. These two notions are reformulated as follows:

Definition 4 (Strict Dominance). *Given a set of measures M, a pattern x_i strictly dominates a pattern x_j with respect to M (denoted by $x_i \gg_M x_j$), iff $\forall m \in M$, $m(x_i) > m(x_j)$.*

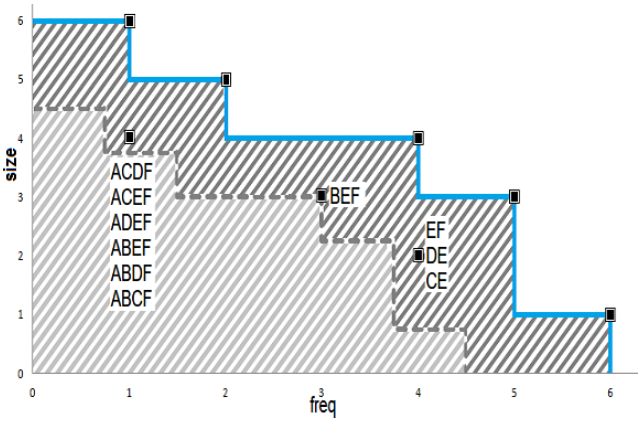


Fig. 4: δ-skypatterns (that are not edge ones) extracted from the example in Table 1.

Definition 5 (Edge-skypattern operator). *Given a pattern set $P \subseteq \mathscr{L}_{\mathscr{I}}$ and a set of measures M, an edge-skypattern of P, with respect to M, is a pattern not strictly dominated in P, with respect to M. The operator $Edge\text{-}Sky(P,M)$ returns all the edge-skypatterns of P with respect to M: $Edge\text{-}Sky(P,M) = \{x_i \in P \mid \nexists x_j \in P, x_j \gg_M x_i\}$*

Given a set of measures M, the edge-skypattern mining problem is thus to evaluate the query $Edge\text{-}Sky(P,M)$. Fig. 3 depicts the 28=7+(4+8+3+4+2) edge-skypatterns extracted from the example in Table 1 for M=$\{freq, size\}$. Obviously, all edge-skypatterns belong to the edge of the dominance area, and seven of them are skypatterns (see Fig. 2).

Proposition 1. *For two patterns x_i and x_j, $x_i \gg_M x_j \implies x_i \succ_M x_j$. So, for a pattern set P and a set of measures M, $Sky(P,M) \subseteq Edge\text{-}Sky(P,M)$.*

3.2 δ-skypatterns

In many cases the user may be interested in skypatterns expressing a trade-off between the measures. The δ-skypatterns address this issue where δ means a percentage of relaxation allowed by the user. Let $0 < \delta \leq 1$.

Definition 6 (δ-Dominance). *Given a set of measures M, a pattern x_i δ-dominates another pattern x_j w.r.t. M (denoted by $x_i \succ_M^{\delta} x_j$), iff $\forall m \in M$, $(1-\delta) \times m(x_i) > m(x_j)$.*

Definition 7 (δ-Skypattern operator). *Given a pattern set $P \subseteq \mathscr{L}_{\mathscr{I}}$ and a set of measures M, a δ-skypattern of P with respect to M is a pattern not δ-dominated in P with respect to M. The δ-skypattern operator $\delta\text{-}Sky(P,M)$ returns all the δ-skypatterns of P with respect to M: $\delta\text{-}Sky(P,M) = \{x_i \in P \mid \nexists x_j \in P : x_j \succ_M^{\delta} x_i\}$.*

The δ-skypattern mining problem is thus to evaluate the query $\delta\text{-}Sky(P,M)$. There are 38 (28+10) δ-skypatterns extracted from the example in Table 1 for M=$\{freq, size\}$ and δ=0.25. Fig. 4 only depicts the 10 δ-skypatterns that are not edge-skypatterns. Intuitively, the δ-skypatterns are close to the edge of the dominance relation, the value of δ is the maximal relative distance between a skypattern and this border.

Proposition 2. *For two patterns x_i and x_j, $x_i \succ_M^{\delta} x_j \implies x_i \gg_M x_j$. So, for a pattern set P and a set of measures M, $Edge\text{-}Sky(P,M) \subseteq \delta\text{-}Sky(P,M)$.*

To conclude, given a pattern set $P \subseteq \mathscr{L}_{\mathscr{I}}$ and a set of measures M, the following inclusions hold: $Sky(P,M) \subseteq Edge\text{-}Sky(P,M) \subseteq \delta\text{-}Sky(P,M)$.

4 Mining (soft-) skypatterns using CP

This section describes how the skypattern and the soft skypattern mining problems can be modeled and solved using Dynamic CSP [22]. A major advantage of this approach is to improve the mining step during the process thanks to constraints dynamically posted and stemming from the current set of the candidate skypatterns. Each time a solution is found, we dynamically post a new constraint leading to reduce the search space. This process stops when we cannot enlarge the forbidden area. Finally, the completeness of our approach is insured by the completeness of the CP solver. The implementation of our approach has been carried out in `Gecode`[1] extending the (CP based) pattern extractor developed by [9].

4.1 CSP and Dynamic CSP

A Constraint Satisfaction Problem (CSP) $P=(\mathscr{X},\mathscr{D},\mathscr{C})$ is defined by:

- a finite set of variables $\mathscr{X}=\{x_1,x_2,\ldots,x_k\}$,
- a domain $\mathscr{D}$, which maps every variable $x_i \in \mathscr{X}$ to a finite set of values $D(x_i)$,
- a finite set of constraints $\mathscr{C}$.

Algorithm 1 [7] shows how a CSP can be solved using a depth-first search. D and C denote respectively the current domains and the current set of constraints. In each node of the search tree, the algorithm branches by assigning values to a variable that is unfixed (line 7). It backtracks when a violation of constraints is found, i.e. at least one domain is empty (line 2). The search is further optimized by carefully choosing the variable that is fixed next (line 5); for instance, heuristics dom/deg selects the variable x_i having the smallest ratio between the size of its current domain and the number of constraints it occurs. The main concept used to speed-up the search is Filtering (constraint propagation) (line 1). Filtering reduces the domains of variables such that the domain remains locally consistent. A solution is obtained (line 9) when each domain $D(x_i)$ is reduced to a singleton and all constraints are satisfied.

Algorithm 1: Constraint-Search(D,C)

```
1 D ← propagate(D,C);
2 if there exists x_i ∈ 𝒳 s.t. D(x_i) is empty then
3 |   return failure;
4 if there exists x_i ∈ 𝒳 s.t. |D(x_i)| > 1 then
5 |   Select x_i ∈ 𝒳 s.t. |D(x_i)| > 1;
6 |   forall v ∈ D(x_i) do
7 |   |   Constraint-Search(D ∪ {x_i -> {v}});
8 else
9 |   output solution D;
```

A Dynamic CSP [22] 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 (addings or removals), domains (value addings or removals), constraints (addings or removals). For our approach, changes are only performed by adding new constraints.

[1] http://www.gecode.org/

Solving such dynamic CSP involves solving a single CSP with additional constraints posted during search. Each time a new solution is found, new constraints $\phi(\mathscr{X})$ are imposed. Such constraints will survive backtracking and state that next solutions should verify both the current set of constraints C and $\phi(\mathscr{X})$. So line 9 of Algorithm 1 becomes:

1 Output solution D;
2 $C \leftarrow C \cup \{\phi(\mathscr{X})\}$

Note that $\mathbb{C}$ is a variable global to all calls to procedure *Constraint-Search*(D,C).

4.2 Mining skypatterns using Dynamic CSP

This section describes our CP approach for mining both skypatterns and soft skypatterns. Constraints on the dominance relation are dynamically posted during the mining process and softness is easily introduced using such constraints.

Variable x will denote the (unknown) skypattern we are looking for. Changes are only performed by adding new constraints (see Section 4.1). So, we consider the sequence $P_1, P_2, ..., P_n$ of CSP where each $P_i = (\{x\}, \mathscr{L}, q_i(x))$ 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 query $q_i(x)$

First, the constraint $closed_M(x)$ states that x must be a closed pattern w.r.t all the measures of M, it allows to reduce the number of redundant patterns[2]. 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, ..., s_i.

Each time the first solution s_i to query $q_i(x)$ is found, we dynamically post a new constraint $\phi_i(x)$ leading to reduce the search space. This process stops when we cannot enlarge the forbidden area (i.e. there exits n s.t. query $q_{n+1}(x)$ has no solution). For skypatterns, $\phi_i(x)$ states that $\neg(s_i \succ_M x)$ (see Definition 2):

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

But, the n extracted patterns s_1, s_2, ..., 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. Filter all patterns $s_i \in S$ that are not skypatterns.

While the number of candidates (n) could be very large (the skypattern mining problem is NP-complete), it remains reasonably-sized in practice for the experiments we conducted (seeTable 2 for the case study in chemoinformatics.)

4.3 Mining soft skypatterns using Dynamic CSP

Soft skypatterns are processed exactly the same way. Each kind of soft skypatterns has its own constraint $\phi_i(x)$ according to its relation of dominance.

[2]The *closed* constraint is used to reduce pattern redundancy. Indeed, **closed skypatterns** make up an exact condensed representation of the whole set of skypatterns [19].

For edge-skypatterns, $\phi_i(x)$ states that $\neg(s_i \gg_M x)$ (see Definition 4):

$$\phi_i(x) \equiv \bigvee_{m \in M} m(s_i) \leq m(x)$$

For δ-skypatterns, $\phi_i(x)$ states that $\neg(s_i \succ_M^{\delta} x)$ (see Definition 6):

$$\phi_i(x) \equiv \bigvee_{m \in M} (1-\delta) \times m(s_i) < m(x)$$

As previously, the n extracted patterns s_1, s_2, …, s_n are not necessarily all soft skypatterns. So, a post processing is required as for skypatterns (see Section 4.2). Mining soft skypatterns is also achieved in a two-steps approach:

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

Once again, the number of candidates (n) remains reasonably-sized in practice for the experiments we conducted (see Table 3).

Pattern variables are set variables represented by their characteristic function with boolean variables. [4, 7] model an unknown pattern x and its associated dataset $\mathscr{T}$ by introducing two sets of boolean variables: $\{X_i \mid i \in \mathscr{I}\}$ where $(X_i = 1) \Leftrightarrow (i \in x)$, and $\{T_t \mid t \in \mathscr{T}\}$ where $(T_t = 1) \Leftrightarrow (x \subseteq t)$. Each set of boolean variables aims at representing the characteristic function of the unknown pattern. For a set of k unknown patterns [9], each pattern x_j is represented by its own set of boolean variables $\{X_{i,j} \mid i \in \mathscr{I}\}$ and $\{T_{t,j} \mid t \in \mathscr{T}\}$.

5 Related Work

Computing skylines is a derivation from the maximal vector problem in computational geometry [13], the Pareto frontier [10] and multi-objective optimization. Since its rediscovery within the database community by [3], several methods have been developed for answering skyline queries [15, 16, 20]. These methods assume that tuples are stored in efficient tree data structures. Alternative approaches have also been proposed to help the user in selecting most significant skylines. For example, [11] measures this significance by means of the number of points dominated by a skyline.

Introducing softness for skylines. [8] have proposed thick skylines to extend the concept of skyline. A thick skyline is either a skyline point p_i, or a point p_j dominated by a skyline point p_i and such that p_j is close to p_i. In this work, the idea of softness is limited to metric semi-balls of radius $\varepsilon>0$ centered at points p_i, where p_i are skylines.

Computing skypatterns is different from computing skylines. Skyline queries focus on the extraction of tuples of the dataset and assume that all the elements are in the dataset, while 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: $O(2^{|\mathscr{I}|})$ instead of $O(|\mathscr{T}|)$ for skylines.

To the best of our knowledge, there are only two works dealing with skypatterns. [19] have proposed an approach taking benefit of theoretical relationships between pattern condensed representations and skypatterns and making the process feasible when the pattern condensed representation can be extracted. Nevertheless, this method can only use a crisp dominance relation. [17] deal with skypatterns from graphs but their technique only maximizes two measures (number of vertices and edge connectivity).

CP for computing the Pareto frontier. [6] has proposed an algorithm that provides the Pareto frontier in a CSP. This algorithm is based on the concept of nogoods[3] and uses spatial data structures (quadtrees) to arrange the set of nogoods. This approach only deals with non-dominated points. Moreover, it cannot be applied for mining skypatterns.

6 Case study: discovering toxicophores

A major issue in chemoinformatics is to establish relationships between chemicals and a given activity (e.g., CL50 is the lethal concentration of a substance required to kill half the members of a tested population after a specified test duration) in ecotoxicity. Chemical fragments[4] which cause toxicity are called *toxicophores* and their discovery is at the core of prediction models in (eco)toxicity [1, 18]. The aim of this present study, which is part of a larger research collaboration with the CERMN Lab, a laboratory of medicinal chemistry, is to investigate the use of softness for discovering toxicophores.

6.1 Experimental protocol

The dataset is collected from the ECB web site[5]. For each chemical, the chemists associate 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 $\mathscr{T}$ consists of 567 chemicals, 372 from the H400 class and 195 from the H402 class. The chemicals are encoded using 1,450 frequent closed subgraphs previously extracted from $\mathscr{T}$[6] with a 1% relative frequency threshold.

In order to discover patterns as candidate toxicophores, we use both measures typically used in contrast mining [14] such as the growth rate since toxicophores are linked to a classification problem with respect to the HSC and measures expressing the background knowledge such as the aromaticity or rigidity because chemists consider that this information may yield promising candidate toxicophores. Our method offers a natural way to simultaneously combine in a same framework these measures coming from various origins. We briefly sketch these measures.

- Growth rate. When a pattern has a frequency which significantly increases from the H402 class to the H400 class, then it stands a potential structural alert related to the toxicity: if a chemical has, in its structure, fragments that are related to a toxic effect, then it is more likely to be toxic. Emerging patterns embody this natural idea by using the growth-rate measure (see Definition 1).

- Frequency. Real-world datasets are often noisy and patterns with low frequency may be artefacts. The minimal frequency constraint ensures that a pattern is representative enough (i.e., the higher the frequency, the better is).

- Aromaticity. Chemists know that the aromaticity 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. We compute the aromaticity of a pattern as the mean of the aromaticity of its chemical fragments.

---

[3] A nogood is a partial or complete assignment of the variables such that there will be no (new) solution containing it.

[4] A fragment denominates a connected part of a chemical structure containing at least one chemical bond.

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

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

	Skypatterns				
		CP+SKY		MICMAC+SKY	
	# of Skypatterns	# 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**	12,126	**21m:40s**

Table 2: Skypattern mining on ECB dataset.

Redundancy is reduced by using **closed skypatterns** which are an exact condensed representation of the whole set of skypatterns (see Footnote 2). We consider four sets of measures: M_1, M_2, M_3 and M_4 (see Table 2). For δ-skypatterns, we consider two values: δ=10% and δ=20%. The extracted skypatterns and soft skypatterns are made of molecular fragments. To evaluate the presence of toxicophores in their description, an expert analysis leads to the identification of well-known environmental toxicophores.

6.2 Performance analysis

This section compares our approach (noted CP+SKY) with MICMAC+SKY, which is the only other method able to mine skypatterns [19]. As our proposal, MICMAC+SKY 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 is applied. Table 2 reports, for each set of measures:

- the number of skypatterns,
- for CP+SKY, the number of candidates (i.e. the number of intermediate patterns, see Section 4.2) and the associated CPU-time,
- for MICMAC+SKY, the number of closed patterns of the condensed representation and the associated CPU-time.

Table 3 reports, for each set of measures:

- the number of edge-skypatterns that are not (hard) skypatterns, the number of candidates and the required CPU-time,
- the number for δ-skypatterns that are not edge-skypatterns, the number of candidates and the required CPU-time.

CP+SKY outperforms MICMAC+SKY in terms of CPU-times (see Table 2). Moreover, the number of candidates generated by our approach remains small compared to the number of closed patterns computed by MICMAC+SKY. Thanks to dynamic constraints, our CP approach enables to drastically reduce the number of candidates. Moreover, increasing the number of measures leads to a larger number of (soft-)skypatterns, particularly for high values of δ. In fact, a pattern rarely dominates all other patterns on the whole set of measures. Nevertheless, in our experiments, the number of soft skypatterns remains reasonably small. For edge-skypatterns, there is a maximum of 144 patterns, while for δ-skypatterns, there is a maximum of 1,724 patterns (for $\delta = 20\%$).

	Edge-Skypatterns			δ-Skypatterns					
				$\delta = 10\%$			$\delta = 20\%$		
	CP+Edge-SKY			CP+δ-SKY					
	# of Edge-skypatterns	# of Candidates	CPU-Time	# of δ-skypatterns	# of Candidates	CPU-Time	# of δ-skypatterns	# of Candidates	CPU-Time
M_1	**24**	1,746	**19m:02s**	**25**	4,204	**20m:48s**	**87**	6,253	**22m:36s**
M_2	**76**	688	**17m:51s**	**354**	1,678	**18m:14s**	**1,670**	2,816	**23m:44s**
M_3	**72**	1,726	**16m:50s**	**352**	4,070	**19m:43s**	**1,654**	6,699	**22m:25s**
M_4	**144**	3,021	**20m:27s**	**385**	6,048	**23m:36s**	**1,724**	8,986	**30m:14s**

Table 3: Soft skypattern mining on ECB dataset.

6.3 Qualitative analysis

In this section, we analyse qualitatively the (soft-)skypatterns by evaluating the presence of toxicophores in their description, according to well-known environmental toxicophores. For M_1=$\{growth\text{-}rate, freq\}$, soft skypatterns enable to efficiently detect well-known toxicophores emphasized by skypatterns, while for M_2=$\{growth\text{-}rate, aromaticity\}$ and M_4=$\{growth\text{-}rate, freq, aromaticity\}$, soft skypatterns enable to discover (new) interesting toxicophores that would not be detected by skypatterns.

- Growth rate and frequency measures (M_1). Only 8 skypatterns are found, and 3 well-known toxicophores are emphasized. Two of them are aromatic compounds, namely the chlorobenzene (p_1) and the phenol rings (p_2). The contamination of water and soil by organic aromatic chemicals is widespread as a result of industrial applications ranging from their use as pesticides, solvents to explosives and dyestuffs. Many of them may bioaccumulate in the food chain and have the potential to be harmful to living systems including humans, animals, and plants. The third one, the organophosphorus moiety (p_3) is a component occurring in numerous pesticides.

Soft skypatterns confirm the trends given by skypatterns. However, the chloro-substituted aromatic rings (e.g. p_4), and the organophosphorus moiety (e.g. p_5) are detected by the edge-skypatterns and by the δ-skypatterns. Indeed, several patterns containing these toxicophores are extracted.

- Growth rate and aromaticity measures (M_2). Figure 6 only reports the distribution of the (soft-)skypatterns for M_2. Soft skypatterns lead to the discovery of several different aromatic rings. In fact, the nature of these chemicals can vary in function of i) the presence/absence of heteroatoms (e.g. N, S), ii) the number of rings, and iii) the presence/absence of substituents.

Edge-skypatterns leads to the extraction of (i) *nitrogen aromatic compounds*: indole (p_1) and benzoimidazole (p_2), (ii) *S-containing aromatic compounds*: benzothiophene (p_3), (iii) *aromatic oxygen compounds*: benzofurane (p_4), and (iv) *polycyclic aromatic hydrocarbons*: naphthalene (p_5). δ-skypatterns complete the list of the aromatic rings which were not found during the extraction of the skypatterns, namely biphenyl (p_6).

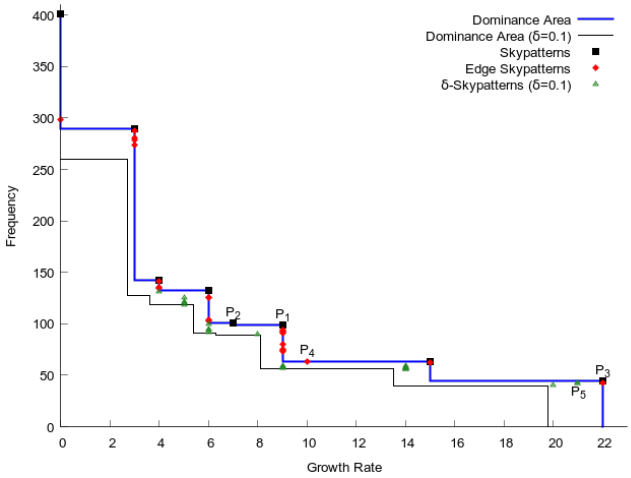


Fig. 5: Analysing the (soft-) skypatterns for M_1.

- Growth rate, frequency and aromaticity measures (M_4)**.** The most interesting results are provided using M_4 (see Figure 7). 21 skypatterns are mined, and several well-known toxicophores are emphasized: the phenol ring (see e_4), the chloro-substituted aromatic ring (see e_3), the alkyl-substituted benzene (see e_2), and the organophosphorus moiety (see P_1). Besides, information dealing with nitrogen aromatic compounds are also extracted (see e_1).

Soft skypatterns enable to mine several exotic aromatic rings (previously discussed), namely nitrogen and S-containing aromatic compounds, polycyclic aromatic hydrocarbons.

Moreover, edge-skypatterns enable to detect more precisely the chloro-substituted aromatic ring and the organophosphorus moiety which are located near P_1. For $\delta \in \{10\%, 20\%\}$, mining the δ-skypatterns leads to the extraction of new several interesting patterns, particularly substituted nitrogen aromatic rings and substituted anilines.

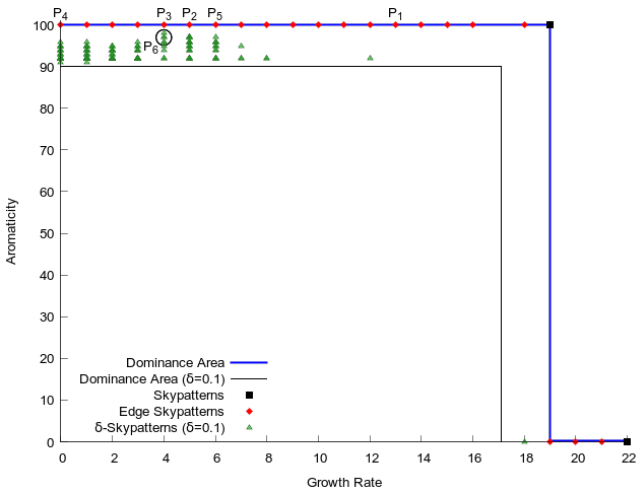


Fig. 6: Analysing the (soft-) skypatterns for M_2.

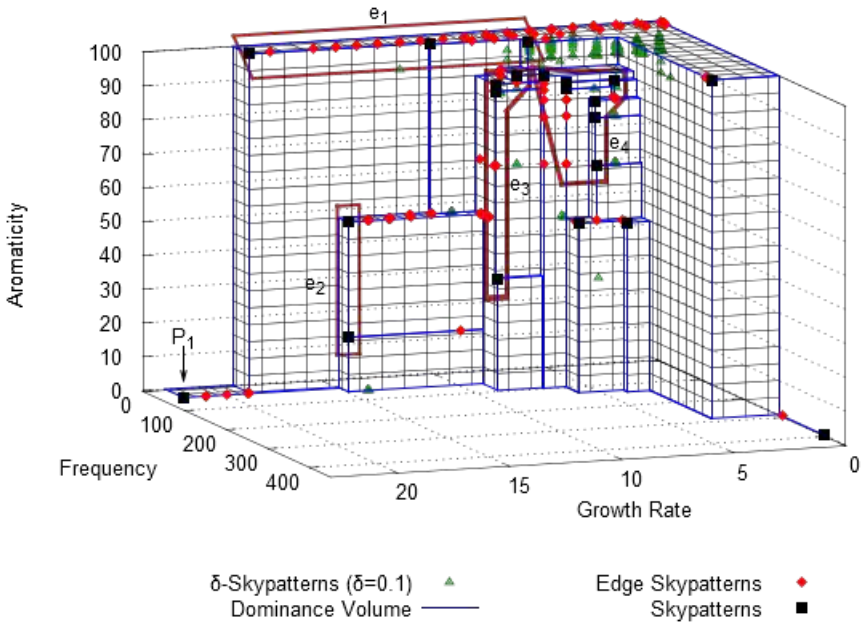


Fig. 7: Analysing the (soft-) skypatterns for M_4.

7 Conclusion

We have introduced the notion of soft skypattern and proposed a flexible and efficient approach to mine skypatterns as well as soft ones thanks to Dynamic CSP. Finally, the relevance and the effectiveness of our approach has been highlighted through a case study in chemoinformatics for discovering toxicophores.

In the future, we would like to study the introduction of softness on other tasks such as clustering, study the contribution of soft skypatterns for recommendation and extend our approach to skycubes. Another direction is to improve the solving stage by designing a one-step method: each time a new solution s_i is found, all candidates that are dominated by s_i can be removed (see Section 4.2). Another idea is to hybridize our CP approach with local search methods to improve the efficiency of the method.

Acknowledgments. We would like to thank Arnaud Soulet (University F. Rabelais of Tours, France), for providing the `MICMAC+SKY` program and his highly valuable comments. This work is partly supported by the ANR (French Research National Agency) funded project FiCoLoFo ANR-10-BLA-0214.

References

1. J. Bajorath and J. Auer. Emerging chemical patterns : A new methodology for molecular classification and compound selection. *J. of Chemical Information and Modeling*, 46:2502–2514, 2006.
2. S. Bistarelli and F. Bonchi. Soft constraint based pattern mining. *Data Knowl. Eng.*, 62(1):118–137, 2007.
3. S. Börzönyi, D. Kossmann, and K. Stocker. The skyline operator. In *17th Int. Conf. on Data Engineering*, pages 421–430. Springer, 2001.
4. L. De Raedt, T. Guns, and S. Nijssen. Constraint programming for itemset mining. In *KDD'08*, pages 204–212. ACM, 2008.
5. L. De Raedt and A. Zimmermann. Constraint-based pattern set mining. In *7th SIAM International Conference on Data Mining*. SIAM, 2007.
6. M. Gavanelli. An algorithm for multi-criteria optimization in csps. In Frank van Harmelen, editor, *ECAI*, pages 136–140. IOS Press, 2002.
7. T. Guns, S. Nijssen, and L. De Raedt. Itemset mining: A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983, 2011.
8. W. Jin, J. Han, and M. Ester. Mining thick skylines over large databases. In *PKDD'04*, pages 255–266, 2004.
9. M. Khiari, P. Boizumault, and B. Crémilleux. Constraint programming for mining n-ary patterns. In *CP'2010*, volume 6308 of *LNCS*, pages 552–567, 2010.
10. H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of ACM*, 22(4):469–476, October 1975.
11. X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In *ICDE 2007*, pages 86–95, 2007.
12. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and K. Discovery*, 1(3):241–258, 1997.
13. J. Matousek. Computing dominances in E^n. *Inf. Process. Lett.*, 38(5):277–278, 1991.
14. P. Kralj Novak, N. Lavrac, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403, 2009.
15. D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1):41–82, 2005.
16. D. Papadias, M. Yiu, N. Mamoulis, and Y. Tao. Nearest neighbor queries in network databases. In *Encyclopedia of GIS*, pages 772–776. 2008.
17. A. N. Papadopoulos, A. Lyritsis, and Y. Manolopoulos. Skygraph: an algorithm for important subgraph discovery in relational graphs. *Data Min. Knowl. Discov.*, 17(1):57–76, 2008.
18. G. Poezevara, B. Cuissart, and B. Crémilleux. Extracting and summarizing the frequent emerging graph patterns from a dataset of graphs. *J. Intell. Inf. Syst.*, 37(3):333–353, 2011.
19. A. Soulet, C. Raïssi, M. Plantevit, and B. Crémilleux. Mining dominant patterns in the sky. In *ICDM*, pages 655–664, 2011.
20. Kian-Lee Tan, Pin-Kwang Eng, and Beng Chin Ooi. Efficient progressive skyline computation. In *VLDB*, pages 301–310, 2001.
21. W. Ugarte, P. Boizumault, S. Loudni, and B. Crémilleux. Soft threshold constraints for pattern mining. In *Discovery Science*, pages 313–327, 2012.
22. G. Verfaillie and N. Jussien. Constraint solving in uncertain and dynamic environments: A survey. *Constraints*, 10(3):253–281, 2005.