Efficient Mining under Flexible Constraints through Several Datasets

Arnaud Soulet, Jiří Kléma, and Bruno Crémilleux

GREYC, CNRS - UMR 6072, Université de Caen
Campus Côte de Nacre, F-14032 Caen Cédex France
{Forename.Surname}@info.unicaen.fr

Abstract. Mining patterns under many kinds of constraints is a key point to successfully get new knowledge. In this paper, we propose an efficient new algorithm Music-dfs which soundly and completely mines patterns with various constraints from large data and takes into account external data represented by several heterogeneous datasets. Constraints are freely built of a large set of primitives and enable to link the information scattered in various knowledge sources. Efficiency is achieved thanks to a new closure operator providing an interval pruning strategy covering in depth the search space. A genomic case study shows both the effectiveness of our approach and the added-value of background knowledge such as free texts or gene ontologies in discovery of meaningful patterns.

1 Introduction

In current scientific, industrial or business data mining applications, the critical need is not to generate data, but to derive knowledge from huge and heterogeneous datasets produced at high throughput. Putting all this data together has become a pressing need for developing environments and tools able to explore and discover new highly valuable knowledge. This involves different challenges, like designing efficient tools to tackle large amount of data and the discovery of patterns of a potential interest for the user through several datasets. By reducing the number of patterns extracted to those of a potential interest given by the user, constraints provide focus on the most promising knowledge. Furthermore, when constraints can be pushed deep inside the mining algorithm, performance is improved, thus making the mining task computationally feasible and resulting in a human-workable output.

This paper addresses the issue of efficient mining under flexible constraints from large binary data combined with several heterogeneous external datasets synthetizing background knowledge (BK). Large datasets are characterized mainly by a large number of columns (i.e., items). This characteristic often encountered in a lot of domains (e.g., bioinformatics, text mining) represents a remarkable challenge. Usual algorithms show difficulties in running on this kind of data due to the exponential search space growth with the number of items. Known level-wise algorithms can fail in mining frequent or constrained patterns in such data [5]. On top of that, the user often would like to integrate BK in the mining process in order to focus on the most plausible patterns. BK is available in relational and literature databases, ontological trees and other sources. Nevertheless, mining in a heterogeneous environment allowing a large set of descriptions at various levels of detail is highly non-trivial. This paper solves the problem by pushing user-defined constraints that may stem both from the mined binary data and the BK summarized in similarity matrices or textual files.

The contribution of this paper is twofold. First we provide an efficient new algorithm Music-dfs which soundly and completely mines constrained patterns from large data and takes into account external data (i.e., several heterogeneous datasets). Except for specific constraints for which tricks like the transposition of data [8, 5] or the use of the extension [4] can be used, levelwise approaches cannot tackle large data due to the huge number of candidates. On the contrary, Music-dfs is based on a depth first search strategy. The key idea is to use a new closure operator enabling an efficient interval pruning strategy (see Section 3). In [3], the authors also benefit from intervals to prune the search space, but their approach is restricted to the conjunction of one monotone constraint and one anti-monotone constraint. The output of Music-dfs is an interval condensed representation: each pattern satisfying the given constraint appears once in the collection of intervals only. Second, we provide a generic framework to mine patterns with a large set of constraints through several heterogeneous datasets like texts or similarity matrices. It is a way to take into account the BK. Section 4 depicts a genomic case study. The biological demands require to mine the expression data with constraints concerning complex relations represented by free texts and gene ontologies. The discovered patterns are likely to encompass interesting and interpretable knowledge.

This papers differs for a double reason from our work in [11]. First, the framework is extended to external data. Second, Music-dfs is deeply different from the prototype used in [11]: Music-dfs integrates primitives to tackle external data and thanks to its strategy to prune the search space (new interval pruning based on prefix-free patterns, see Section 3), it is able to mine large data. Section 4 demonstrates the practical effectiveness of Music-dfs in a genomic case study and shows that other prototypes (including the prototype presented in [11]) fail. To the best of our knowledge, there is no other constraint-based tool to efficiently discover patterns from large data under a broad set of constraints linking the information distributed in various knowledge sources.

This paper is organized as follows. Section 2 defines our framework to mine patterns satisfying constraints defined over several kinds of datasets. In Section 3, we present the theoretical essentials that underlie the efficiency of Music-dfs and we provide its main features. Experiments showing the efficiency of Music-dfs and the cross-fertilization between several sources of information related to the genomic area are given in Section 4.

2 Defining Constraints Through Several Datasets

Usual data-mining tasks can rarely be represented by a single binary dataset. Often it is necessary to connect knowledge scattered in several heterogeneous sources. In constraint-based mining, the constraints should effectively link different datasets and knowledge types. For instance, in the domain of genomics, biologists are interested in constraints both on synexpression groups and common characteristics of the genes and/or biological situations concerned. Such constraints require to tackle both transcriptome data (often provided in a transactional format) and literature databases. This section presents our framework (and the declarative language) enabling the user to set varied and meaningful constraints. We describe our framework by starting from a genomic example.

Let us consider the genomic mining context given in Figure 1. Firstly, this context is made up of a boolean dataset also called internal data (or transcriptome data) where the items correspond to genes, the transactions represent biological situations. Secondly, external data (a similarity matrix and textual resources) are considered. They summarize the BK that contains various information on items (i.e., genes). This knowledge is transformed into a similarity matrix and a set of texts. Each field of the triangular matrix $s_{ij} \in [0,1]$ gives a similarity measure between the items i and j (or transactions respectively). The textual dataset provides a description of genes. Each row of this dataset contains a list of phrases characterizing the given gene. The mined patterns are composed of items of the internal data, the external data are used to further specify constraints in order to focus on meaningful patterns. In other words, the constraints may stem from all the datasets (see the example of q in Figure 1, Section 4 provides another q').

Internal data

Boolean matrix $\mathcal{D}$

Situations	Genes
s_1	A E F
s_2	B C D
s_3	A B C D E F
s_4	A B C D

External data

Similarity matrix

	A	B	C	D	E	F
A		.07	?	?	.2	0
B			.06	?	?	0
C				.07	.05	.04
D					.03	.1
E						?

Textual data

A	'metal ion binding' 'transcription factor'
B	'serine-type peptidase activity' 'proteolysis'
C	'DNA binding' 'metal ion binding'
D	'ATP binding' 'nucleotide binding'
E	'proteolysis'
F	'ATP binding' 'metal ion binding'

freq, length,...
sumsim, svmsim,...
regexp

$$\begin{aligned} q(X) \equiv\ & freq(X) \times length(X) \geq 24 && \text{(a)}\\ & \wedge\, length(regexp(X, '*ribosom*', \texttt{TEXT\_terms})) \leq 1 && \text{(b)}\\ & \wedge\, svsim(X, \texttt{TEXT})/(svsim(X, \texttt{TEXT}) + mvsim(X, \texttt{TEXT})) \geq 0.7 && \text{(c)}\\ & \wedge\, sumsim(X, \texttt{TEXT})/svsim(X, \texttt{TEXT}) \geq 0.025 && \text{(d)} \end{aligned}$$

Fig. 1. Example of a toy (genomic) mining context and a constraint.

Let $\mathcal{I}$ be a set of items, a pattern is a non-empty subset of $\mathcal{I}$. $\mathcal{D}$ is a boolean matrix composed of patterns usually called transactions. The constraint-based mining task aims to discover all the patterns present in $\mathcal{D}$ and satisfying a constraint q. A pattern X is present in $\mathcal{D}$ whenever it is included in one transaction of $\mathcal{D}$ at least. One originality of our framework lies in its flexibility. Constraints are freely built of a large set of primitives representing an integrative, iterative and rich query language.

Table 1 provides the meaning of the primitives involved in q and also the constraints used in Section 4. As primitives on external data are derived from different datasets, the dataset makes another parameter of the primitive (it is not present in Table 1 to alleviate the writing). The first part (a) of q addresses the internal data and means that the biologist is interested in patterns having a satisfactory size (i.e., a ***minimal area***). Indeed, $area(X) = freq(X) \times length(X)$ is the product of the frequency of X and its length and means that the pattern must cover a minimum number of situations and contain a minimum number of genes. The other parts deal with the external data: (b) is used to discard ribosomal patterns (one gene exception per pattern is allowed), (c) to avoid patterns with prevailing items of an unknown function and (d) to ensure a minimal average similarity. Table 1 also indicates the values of these primitives in the context of Figure 1. Our framework supports a large set of primitives, other examples of primitives are $\{\wedge, \vee, \neg, <, \leq, \subset, \subseteq, +, -, \times, /, sum, max, min, \cup, \cap, \setminus\}$. The only property which is required on the primitives to belong to our framework is a property of monotonicity according to each variable of a primitive [11]. The constraints of this framework are called ***primitive-based constraints***. Let us recall that the primitives and the constraints defined in [11] only address one boolean data set.

primitives		values
	Boolean matrix	
$freq(X)$ $length(X)$	frequency of X length of X	$freq(ABC) = 2$ $length(ABC) = 3$
	Textual data	
$regexp(X, RE)$	items of X whose associated phrases match the regular expression RE	$regexp(ABC, '* ion *')$ $= AC$
	Similarity matrix	
$sumsim(X)$	similarity sum over the set of item pairs of X	$sumsim(ABC) = 0.13$
$svsim(X)$	number of stated item pairs belonging to X	$svsim(ABC) = 2$
$mvsim(X)$	number of missing item pairs belonging to X	$mvsim(ABC) = 1$
$insim(X, min, max)$	number of item pairs whose similarity lies between min and max	$insim(ABC, 0.07, 1) = 1$

Table 1. Examples of primitives and their values in the data mining context of Figure 1.

3 Music-dfs Tool

This section presents the Music-dfs tool (Mining with a User-SpecifIed Constraint, Depth-First Search approach) which benefits from the primitive-based constraint presented in the previous section. Efficiency is achieved thanks to the exploitation of the primitive and constraint properties. We start by giving the key idea of the safe pruning process based on intervals.

3.1 Main features of the interval pruning

We give the intuition of the pruning process performed by Music-dfs. The key idea is to exploit properties of the monotonicity of the primitives (see Section 2) on the bounds of intervals to prune them. This new kind of pruning is called *interval pruning*. Given two patterns $X \subseteq Y$, the interval $[X, Y]$ corresponds to the set $\{Z \subseteq \mathcal{I} \mid X \subseteq Z \subseteq Y\}$. Figure 2 depicts an example with the interval $[AB, ABCD]$ and the values of the primitives *sumsim* and *svsim*.

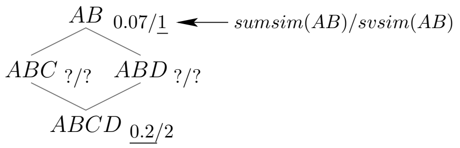

Fig. 2. Illustration of the interval pruning.

Assume the constraint $sumsim(X)/svsim(X) \geq 0.25$. As the values associated to the similarities are positive, $sumsim(X)$ is an increasing function according X. Thus $sumsim(ABCD)$ is the highest *sumsim* value for the patterns in $[AB, ABCD]$. Similarly, all the patterns of this interval have a higher $svsim(X)$ value than $svsim(AB)$. Thereby, each pattern in $[AB, ABCD]$ has its average similarity lower or equal than $sumsim(ABCD)/svsim(AB) = 0.2/1$. As this fraction does not exceed 0.25, no pattern of $[AB, ABCD]$ can satisfy the constraint and this interval can be pruned. We say that this pruning is ***negative*** because no pattern satisfies the constraint. In the same way, if the values of proper combinations of the primitives on the bounds of an interval $[X, Y]$ show that all the patterns in $[X, Y]$ satisfy the constraint, then $[X, Y]$ is also pruned and this pruning is named ***positive***. For instance, assuming that $sumsim(AB)/svsim(ABCD) \geq 0.02$, then all the patterns in $[AB, ABCD]$ satisfy the constraint.

In a more formal way, this approach is performed by two interval pruning operators $\lfloor . \rfloor$ and $\lceil . \rceil$ introduced in [11] but only for primitives handling boolean data. The main idea of these operators is to recursively decompose the constraint to take into account the monotone properties of the primitives and then to safely negatively or positively prune intervals as depicted above. This process straightforwardly works with all the primitives tackling several kinds of datasets, this highlights the generic properties of our framework. Thereby, all the parts of the constraint q are pushed into the mining step. The next section indicates how the intervals are built.

3.2 Interval condensed representation

As indicated in the introduction, levelwise algorithms are not suitable to mine datasets with a large number of items due to the huge number of candidates growing exponentially according to the number of items. We adopt a depth-first search strategy instead to enumerate the candidate patterns and to avoid subsequent memory failures. We introduce a new and specific closure operator based on a prefix ordering relation $\preceq$. We show that this closure operator is on the core of the interval condensed representation (Theorem 1) leading to an efficient pruning strategy covering in depth the search space.

The prefix ordering relation $\preceq$ takes into account an arbitrary order over items $A < B < C < \ldots$ as done in [10]. We say that an ordered pattern $X = x_1 x_2 \ldots x_n$

(i.e., $\forall i < j$, we have $x_i < x_j$) is a prefix of an ordered pattern $Y = y_1 y_2 \dots y_m$ and note $X \preceq Y$ iff we have $n \leq m$ and $\forall i \in \{1, \dots, n\}$, $x_i = y_i$. For instance, $ADC \npreceq AD$ because $ADC = ACD$ and AD is not a prefix of ACD.

Definition 1 (Prefix-closure). *The prefix-closure of a pattern X, denoted $\mathbf{cl}_{\preceq}(X)$, is the pattern $\{a \in \mathcal{I} | \exists Y \subseteq X$ such that $Y \preceq Y \cup \{a\}$ and $freq(Ya) = freq(Y)\}$.*

The pattern $\mathbf{cl}_{\preceq}(X)$ gathers together the items occurring in all the transactions containing $Y \subseteq X$ such that Y is a prefix of $Y \cup \{a\}$. The fixed points of operator $\mathbf{cl}_{\preceq}$ are named the *prefix-closed patterns*. Let us illustrate this definition on our running example (cf. Figure 1). The pattern ABC is not a prefix-closed pattern because ABC is a prefix of $ABCD$ and $freq(ABCD) = freq(ABC)$. We straightforwardly deduce that any pattern and its prefix-closure have the same frequency. For instance, as $\mathbf{cl}_{\preceq}(ABC) = ABCD$, $freq(ABC) = freq(ABCD) = 2$. We show now the property of closure of $\mathbf{cl}_{\preceq}$:

Property 1 (Closure operator) *The prefix-closure operator $\mathbf{cl}_{\preceq}$ is a closure operator.*

Proof. *Extensivity:* Let X be a pattern and $a \in X$. We have $\{a\} \subseteq X$ and obviously, $a \preceq a$ and $freq(a) = freq(a)$. Then, we obtain that $a \in \mathbf{cl}_{\preceq}(X)$ and $\mathbf{cl}_{\preceq}$ is extensive. *Isotony:* Let $X \subseteq Y$ and $a \in \mathbf{cl}_{\preceq}(X)$. There exists $Z \subseteq X$ such that $Z \preceq Za$ and $freq(Za) = freq(Z)$. As we also have $Z \subseteq Y$ (and $freq(Za) = freq(Z)$), we obtain that $a \in \mathbf{cl}_{\preceq}(Y)$ and conclude that $\mathbf{cl}_{\preceq}(X) \subseteq \mathbf{cl}_{\preceq}(Y)$. *Idempotency:* Let X be a pattern. Let $a \in \mathbf{cl}_{\preceq}(\mathbf{cl}_{\preceq}(X))$. There exists $Z \subseteq \mathbf{cl}_{\preceq}(X)$ such that $freq(Za) = freq(Z)$ with $Z \preceq Za$. As $Z \subseteq \mathbf{cl}_{\preceq}(X)$, for all $a_i \in Z$, there is $Z_i \subseteq X$ such that $freq(Z_i a_i) = freq(Z_i)$ with $Z_i \preceq Z_i a_i$. We have $\bigcup_i Z_i \preceq \bigcup_i Z_i a$ and $freq(\bigcup_i Z_i) = freq(\bigcup_i Z_i a)$ (because $freq(\bigcup_i Z_i) = freq(Z)$). As the pattern $\bigcup_i Z_i \subseteq X$, a belongs to $\mathbf{cl}_{\preceq}(X)$ and then, $\mathbf{cl}_{\preceq}$ is idempotent. □

Property 1 is important because it enables to infer results requiring the properties of a closure operator. First, this new prefix-closure operator designs *equivalence classes* through the lattice of patterns. More precisely, two patterns X and Y are equivalent iff they have the same prefix-closure (i.e., $\mathbf{cl}_{\preceq}(X) = \mathbf{cl}_{\preceq}(Y)$). Of course, as $\mathbf{cl}_{\preceq}$ is idempotent, the maximal (w.r.t. $\subseteq$) pattern of a given equivalence class of X corresponds to the prefix-closed pattern $\mathbf{cl}_{\preceq}(X)$. Conversely, we call *prefix-free patterns* the minimal (w.r.t. $\subseteq$) patterns of equivalence classes. Second, closure properties enable to prove that the prefix-freeness is an anti-monotone constraint (see Property 2 in the next section).

Contrary to the equivalence classes defined by the Galois closure [2, 9], equivalence classes provided by $\mathbf{cl}_{\preceq}$ have a unique prefix-free pattern. This allows to prove that a pattern belongs to one interval only and provides the important result on the interval condensed representation (cf. Theorem 1). This result cannot be achieved without the new closure operator. Lemma 1 indicates that any equivalence class has a unique prefix-free pattern:

Lemma 1 (Prefix-freeness operator). *Let X be a pattern, there exists an unique minimal (w.r.t. $\subseteq$) pattern, denoted $\mathbf{fr}_{\preceq}(X)$, in its equivalence class.*

Proof. Supposing that X and Y are two minimal patterns of the same equivalence class: we have $\mathbf{cl}_{\preceq}(X) = \mathbf{cl}_{\preceq}(Y)$. As X and Y are different, there exists $a \in X$ such that $a \notin Y$ and $a \leq min_{\leq}\{b \in Y \backslash X\}$ (or we invert X and Y). As X is minimal, no pattern $Z \subseteq X \cap Y$ satisfies that $Z \preceq Za$ and $freq(Za) = freq(Z)$. Besides, for all Z such that $Y \cap X \subset Z \subset Y$, we have $Z \npreceq Za$ because a is smaller than any item of $Y \backslash X$. So, a does not belong to $\mathbf{cl}_{\preceq}(Y)$ and then, we obtain that $\mathbf{cl}_{\preceq}(X) \neq \mathbf{cl}_{\preceq}(Y)$. Thus, we conclude that any equivalence class exactly contains one prefix-free pattern. □

Lemma 1 means that the operator $\mathbf{fr}_{\preceq}$ links a pattern X to the minimal pattern of its equivalence class, i.e. $\mathbf{fr}_{\preceq}(X)$. X is prefix-free iff $\mathbf{fr}_{\preceq}(X) = X$. Any equivalence class corresponds to an interval delimited by one prefix-free pattern and its prefix-closed pattern (i.e., $[\mathbf{fr}_{\preceq}(X), \mathbf{cl}_{\preceq}(X)]$). For example, AB (resp. $ABCD$) is the prefix-free (resp. prefix-closed) pattern of the equivalence class $[AB, ABCD]$.

Now let us show that the whole collection of the intervals formed by all the prefix-free patterns and their prefix-closed patterns provides an *interval condensed representation* where each pattern X is present only once in the set of intervals.

Theorem 1 (Interval condensed representation). *Each pattern X present in the dataset is included in the interval $[\mathbf{fr}_{\preceq}(X), \mathbf{cl}_{\preceq}(X)]$. Besides, the number of these intervals is less than or equal to the number of patterns.*

Proof. Let X be a pattern and $R = \{[\mathbf{fr}_{\preceq}(X), \mathbf{cl}_{\preceq}(X)] | freq(X) \geq 1\}$. Lemma 1 proves that X is exactly contained in $[\mathbf{fr}_{\preceq}(X), \mathbf{cl}_{\preceq}(X)]$. The latter is unique. As X belongs to R by definition, we conclude that R is a representation of any pattern. Now, the extensivity and the idempotency of prefix-closure operator $\mathbf{cl}_{\preceq}$ ensure that $|R| \leq |\{X \subseteq \mathcal{I}$ such that $freq(X) \geq 1\}|$. Thus, Theorem 1 is right. □

In the worst case the size of the condensed representation is the number of patterns (each pattern is its own prefix-free and its own prefix-closed pattern). But, in practice, the number of intervals is low compared to the number of patterns (in our running example, only 23 intervals sum up the 63 present patterns).

The condensed representation highlighted by Theorem 1 differs from the condensed representations of frequent patterns based on the Galois closure [2, 9]: in this last case, intervals are described by a free (or key) pattern and its Galois closure and a frequent pattern may appear in several intervals. We claim that the presence of a constraint pattern in a single interval brings meaningful advantages: the mining is more efficient because each pattern is tested at most once and this improves the synthesis of the output of the mining process and facilitates its analysis by the end-user. The next section shows that by combining this condensed representation and the interval pruning operators, we get an interval condensed representation of primitive-based constrained patterns.

3.3 Mining primitive-based constraints in large datasets

When running, Music-dfs enumerates all the intervals sorted in a lexicographic order and checks whether they can be pruned as proposed in Section 3.1. The enumeration benefits from the anti-monotonicity property of the prefix-freeness (cf. Property 2). The memory requirements only grow linearly with the number of items and the number of transactions.

Property 2 *The prefix-freeness is an anti-monotone constraint (w.r.t. $\subseteq$).*

The proof of Property 2 is very similar with those of the usual freeness [2, 9]. In other words, the anti-monotonicity ensures us that once we know that a pattern is not prefix-free, any superset of this pattern is not prefix-free anymore [1, 7]. Algorithms 1 and 2 give the sketch of Music-dfs.

Algorithm 1 GlobalScan

```
Input: A prefix-pattern $X$, a primitive based constraint $q$ and a dataset $\mathcal{D}$
Output: Interval condensed representation of constrained patterns having $X$ as prefix
1: if $\neg PrefixFree(X)$ then return $\emptyset$    // anti-monotone pruning
2: return LocalScan($[X, \mathbf{cl}_{\preceq}(X)], q, \mathcal{D}$)    // local mining
   $\cup \bigcup\{$GlobalScan$(Xa, q, \mathcal{D}) | a \in \mathcal{I} \wedge a \geq \max_{\leq} X\}$    // recursive enumeration
```

Algorithm 2 LocalScan

```
Input: An interval $[X, Y]$, a primitive based constraint $q$ and a dataset $\mathcal{D}$
Output: Interval condensed representation of constrained patterns of $[X, Y]$
1: if $\lfloor q \rfloor \langle X, Y \rangle$ then return $\{[X, Y]\}$    // positive interval pruning
2: if $\neg \lceil q \rceil \langle X, Y \rangle$ then return $\emptyset$    // negative interval pruning
3: if $q(X)$ then return $[X, X] \cup \bigcup\{$LocalScan$([Xa, \mathbf{cl}_{\preceq}(Xa)], q, \mathcal{D}) | a \in Y \backslash X\}$
4: return $\bigcup\{$LocalScan$([Xa, \mathbf{cl}_{\preceq}(Xa)], q, \mathcal{D}) | a \in Y \backslash X\}$    // recursive division
```

Music-dfs scans the whole search space by running GlobalScan on each item of $\mathcal{I}$. GlobalScan recursively performs a depth-first search and stops whenever a pattern is not prefix-free (Line 1, GlobalScan). For each prefix-free pattern X, it computes its prefix-closed pattern and builds $[X, \mathbf{cl}_{\preceq}(X)]$ (Line 2, GlobalScan). Then, LocalScan tests this interval by using the operators $\lfloor . \rfloor$ and $\lceil . \rceil$ informally presented in Section 3.1. If the interval pruning can be performed, the interval is selected (positive pruning, Line 1 from LocalScan) or rejected (negative pruning, Line 2 from LocalScan). Otherwise, the interval is explored by recursively dividing it (Line 3 or 4 from LocalScan). The decomposition of the intervals is done so that each pattern is considered only once. The next theorem provides the correctness of Music-dfs:

Theorem 2 (Correctness). Music-dfs *mines soundly and completely all the patterns satisfying q by means of intervals.*

Proof. Property 2 ensures us that Music-dfs enumerates all the interval condensed representation. Thereby, any pattern is considered (Theorem 1) individually or globally with the safe pruning stemmed from to the interval pruning (see Section 3.1). □

4 Mining Constraint Patterns from Genomic Data

This section depicts the effectiveness of our approach on a genomic case study. We experimentally show two results. First, the usefulness of the interval pruning strategy of MUSIC-DFS (the other prototypes fail for such large data, cf. Section 4.2). Second, BK enables to focus automatically on the most plausible candidate patterns (cf. Section 4.3). This underlines the need to mine constrained patterns by taking into account external data.

4.1 Gene expression data and background knowledge

In this experiment we deal with the SAGE (Serial Analysis of Gene Expression) [12] human expression data downloaded from the NCBI website (`www.ncbi.nlm.nih.gov`). The final binary dataset contains 11082 genes tested in 207 biological situations, each gene can be either over-expressed in the given situation or not. The biological details regarding gene selection, mapping and binarization can be seen in [6].

BK available in literature databases, biological ontologies and other sources is used to help to focus automatically on the most plausible candidate patterns. We have experimented with the gene ontology (GO) and free-text data. First, the available gene databases were automatically searched and the records for each gene were built (around two thirds of genes have non-empty records, there is no information available for the rest of them). Then, various similarity metrics among the gene records were proposed and calculated. The gene records were also simplified to get a condensed textual description. The details on text mining, gene ontologies and similarities are in [6].

4.2 Efficiency of MUSIC-DFS

Let us show the necessity of the depth-first search and usefulness of the interval pruning strategy of MUSIC-DFS. All the experiments were conducted on a 2.2 GHz Pentium IV processor with Linux operating system and 3GB of RAM memory.

The first experiment highlights the importance of the depth-first search. We consider the constraint addressing patterns having an $area \geq 70$ (the minimal area constraint has been introduced in Section 2) and appearing at least 4 times in the dataset. MUSIC-DFS only spends 7sec to extract 212 constrained patterns. In comparison, for the same binary dataset, the levelwise approach[1] presented in [11] fails after 963sec whenever it contains more than 3500 genes. Indeed, the candidate patterns necessary to build the output do not fit in memory.

Comparison with prototypes coming from the FIMI repository (`fimi.cs.helsinki.fi`) shows that efficient implementations like KDCI, LCM (ver. 2), COFI or Borgelt's APRIORI fail with this binary dataset to mine frequent patterns occuring at least 4 times. Borgelt's ECLAT and AFOPT which are depth-first approaches, are able to mine with this frequency constraint. But they require a

[1] We do not use external data because this version does not deal with external data.

post-processing step for taking into account other constraints than the frequency (e.g., area, similarity-based constraints).

The next experiment shows the great role of the interval pruning strategy. For this purpose, we compare Music-dfs with its modification that does not prune. The modification, denoted Music-dfs-filter, mines all the patterns that satisfy the frequency threshold first, the other primitives are applied in the post-processing step. We use two typical constraints needed in the genomic domain and requiring the external data. These constraints and the time comparison between Music-dfs and Music-dfs-filter are given in Figure 3. The results show that post-processing is feasible until the frequency threshold generates reasonable pattern sets. For lower frequency thresholds, the number of patterns explodes and large intervals to be pruned appear. The interval pruning strategy decreases runtime and scales up much better than the comparative version without interval pruning and Music-dfs becomes in the order of magnitude faster.

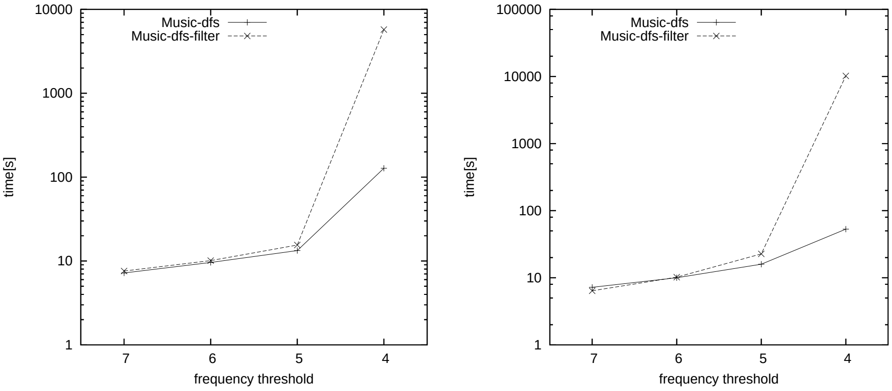

Fig. 3. Efficiency of interval pruning with decreasing frequency threshold. The left image deals with the constraint $freq(X) \geq thres \wedge lenght(X) \geq 4 \wedge sumsim(X)/svsim(X) \geq 0.9 \wedge svsim(X)/(svsim(X) + mvsim(X)) \geq 0.9$. The right image deals with the constraint $freq(X) \geq thres \wedge length(regexp(X,' *ribosom*', \texttt{GO\_terms})) = 0$.

4.3 Use of background knowledge to mine plausible patterns

This genomic case study demonstrates that constraints coming from the BK can reduce the number of patterns, they can express various kinds of interest and the patterns that tend to reappear are likely to be recognized as interesting by an expert. Such an approach requires a tool dealing with external data.

Let us consider all the patterns having a satisfactory size which is translated by the constraint $area \geq 20$[2]. We get nearly half a million different patterns that are joined into 37852 intervals. Although the intervals prove to provide a good

[2] This threshold has been settled by statistical analysis of random datasets having the same properties as the original SAGE data. First spurious patterns start to appear for this threshold area.

condensation, the manual search through this set is obviously infeasible as the interpretation of patterns is not trivial and asks for frequent consultations with medical databases. The biologists prefer sets with tens of patterns/intervals only.

Increasing the threshold of the area constraint to get a reasonable number of patterns is rather counter-productive. The constraint $area \geq 75$ led to a small but uniform set of 56 patterns that was flooded by the ribosomal proteins which generally represent the most frequent genes in the dataset. Biologists rated these patterns as valid but uninteresting.

The most valuable patterns expected by biologists have non-trivial size containing genes and situations whose characteristics can be generalized, connected, interpreted and thus transformed into knowledge. To get such patterns, constraints based on the external data have to be added to the minimal area constraint just like in the constraint q given in Section 2. It joins the minimal area constraint with background constraints coming from the NCBI textual resources (gene summaries and adjoined PubMed abstracts). There are 46671 patterns satisfying the minimal area constraint (the part (a) of the constraint q), but only 9 satisfy q. This shows the efficiency of reduction of patterns brought by the BK.

A cross-fertilization with other external data is obviously favourable. So, we use the constraint q' which is similar to q, except that the functional Gene Ontology is used instead of NCBI textual resources and a similarity constraint is added (part (e) of q').

$$
\begin{aligned}
q'(X) \equiv\ & area(X) \geq 24 && \text{(a)}\\
& \wedge\, length(regexp(X, '*ribosom*', \texttt{GO\_terms})) \leq 1 && \text{(b)}\\
& \wedge\, svsim(X, \texttt{GO})/(svsim(X, \texttt{GO}) + mvsim(X, \texttt{GO})) \geq 0.7 && \text{(c)}\\
& \wedge\, sumsim(X, \texttt{GO})/svsim(X, \texttt{GO}) \geq 0.025 && \text{(d)}\\
& \wedge\, insim(X, 0.5, 1, \texttt{GO})/svsim(X, \texttt{GO}) \geq 0.6 && \text{(e)}
\end{aligned}
$$

Only 2 patterns satisfy q'. A very interesting observation is that the pattern[3] that was identified by the expert as one of the "nuggets" provided by q is also selected by q'. This pattern can be verbally characterized as follows: it consists of 4 genes that are over-expressed in 6 biological situations, it contains at most one ribosomal gene, the genes share a lot of common terms in their descriptions as well as they functionally overlap, at least 3 of the genes are known (have a non-empty record) and all of the biological situations are medulloblastomas which are very aggressive brain tumors in children. The example demonstrates two different ways to reach a compact and meaningful output that can be easily human surveyed.

5 Conclusion

Knowledge discovery from a large binary dataset supported by heterogeneous BK is an important task. We have proposed a generic framework to mine patterns with a large set of constraints linking the information scattered in various knowledge

[3] The pattern consists of 4 genes KHDRBS1 NONO TOP2B FMR1 over-expressed in 6 biological situations BM_P019 BM_P494 BM_P608 BM_P301 BM_H275 BM_H876. BM stands for brain medulloblastoma.

sources. We have presented an efficient new algorithm MUSIC-DFS which soundly and completely mines such constrained patterns. Effectiveness comes from an interval pruning strategy based on prefix free patterns. To the best of our knowledge, there is no other contraint-based tool able to solve such constraint-based tasks.

The genomic case study demonstrates that our approach can handle large datasets. It also shows practical utility of the flexible framework integrating heterogeneous knowledge sources. The language of primitives and compounds applied to a wide spectrum of genomic data results in constraints formalizing viable notion of interestingness.

Acknowledgements. The authors thank the CGMC Laboratory (CNRS UMR 5534, Lyon) for providing the gene expression database and many valuable comments. This work has been partially funded by the ACI "masse de données" (French Ministry of research), Bingo project (MD 46, 2004-07).

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 432–444, 1994.

[2] J. F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery journal*, 7(1):5–22, 2003.

[3] C. Bucila, J. Gehrke, D. Kifer, and W. M. White. Dualminer: A dual-pruning algorithm for itemsets with constraints. *Data Min. Knowl. Discov.*, 7(3):241–272, 2003.

[4] C. Hébert and B. Crémilleux. Mining frequent δ-free patterns in large databases. In A. Hoffmann, H. Motoda, and T. Scheffer, editors, *proceedings of the 8th International Conference on Discovery Science (DS'05)*.

[5] B. Jeudy and F. Rioult. Database transposition for constrained (closed) pattern mining. In *KDID*, volume 3377 of *Lecture Notes in Computer Science*, pages 89–107. Springer, 2004.

[6] J. Kléma, A. Soulet, B. Crémilleux, S. Blachon, and O. Gandrillon. Mining plausible patterns from genomic data. In *CBMS 2006 (to appear)*, Salt Lake City, Utah, 2006.

[7] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[8] F. Pan, G. Cong, A. K. H. Tung, Y. Yang, and M. J. Zaki. CARPENTER: finding closed patterns in long biological datasets. In *proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'03)*, pages 637–642, Washington, DC, USA, 2003. ACM Press.

[9] N. Pasquier, Y. Bastide, T. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. *Lecture Notes in Computer Science*, 1540:398–416, 1999.

[10] J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent item sets with convertible constraints. In *ICDE*, pages 433–442. IEEE Computer Society, 2001.

[11] A. Soulet and B. Crémilleux. An efficient framework for mining flexible constraints. In *PAKDD*, volume 3518 of *Lecture Notes in Computer Science*, pages 661–671. Springer, 2005.

[12] V. Velculescu, L. Zhang, B. Vogelstein, and K. Kinzler. Serial analysis of gene expression. *Science*, 270:484–7, 1995.