# A Unifying Framework for Mining Approximate Top-*k* Binary Patterns

Claudio Lucchese, Salvatore Orlando, and Raffaele Perego

**Abstract**—A major mining task for binary matrixes is the extraction of approximate top-$k$ patterns that are able to concisely describe the input data. The top-$k$ pattern discovery problem is commonly stated as an optimization one, where the goal is to minimize a given cost function, see the accuracy of the data description. In this work, we review several greedy algorithms, and discuss PaNDa$^+$, an algorithmic framework able to optimize different cost functions generalized into a unifying formulation. We evaluated the goodness of the algorithm by measuring the quality of the extracted patterns. We adapted standard quality measures to assess the capability of the algorithm to discover both the items and transactions of the patterns embedded in the data. The evaluation was conducted on synthetic data, where patterns were artificially embedded, and on real-world text collection, where each document is labeled with a topic. Finally, in order to qualitatively evaluate the usefulness of the discovered patterns, we exploited PaNDa$^+$ to detect overlapping communities in a bipartite network. The results show that PaNDa$^+$ is able to discover high-quality patterns in both synthetic and real-world datasets.

**Index Terms**—Mining methods and algorithms, 0-1 data, approximate top-$k$ patterns, communities in bipartite networks, MDL

## 1 Introduction

Binary matrixes are derived from several typologies of datasets, collected in diverse application domains. Without loss of generality, we can think of a binary matrix as a representation of a transactional database, composed of a multi-set of transactions (matrix rows), each including a set of items (matrix columns). An *approximate pattern* extracted from a binary matrix thus corresponds to a pair of sets, items and transactions, where the items of the former set are approximately included in all the transactions of the latter set.

Top-$k$ pattern mining is an alternative approach to pattern enumeration. It aims at discovering the (small) set of $k$ patterns that best *describes*, or *models*, the input data. State-of-the-art algorithms differ in the formalization of the concept of *dataset description*. The goodness of a description is measured with some cost function, and the top-$k$ mining task is casted into an optimization of such cost. In most of such formulations, the problem is demonstrated to be NP-hard, and therefore greedy strategies are adopted. At each iteration, the pattern that best optimizes the given cost function is added to the solution. This is repeated until $k$ patterns have been found or until it is not possible to improve the cost function.

In this paper we analyze in depth three state-of-the-art algorithms for approximate top-$k$ pattern mining from binary data: Asso [1], Hyper+ [2] and PaNDa [3]. Our analysis explores the cost functions used by such greedy algorithms and focuses on the evaluation of the extracted patterns. We show that the cost functions adopted by all these algorithms can be generalized into a unique formulation. We thus extend PaNDa by plugging into its framework such generalized formulation, which makes it possible to greedily mine approximate patterns according to several cost functions, including the ones based on the Minimum Description Length (MDL) principle [4]. Finally, we embed into this framework noise constraints inspired to [5] in order to improve the algorithm's greedy heuristics. We name the resulting algorithm PaNDa$^+$.

Concerning the evaluation methodology, we observe that state-of-the-art approaches measure the goodness of the discovered patterns by their capability of minimizing the same cost function optimized by the greedy algorithm. This simple assessment methodology captures the effectiveness of the greedy strategy rather than the quality of the extracted patterns. In this paper, we estimate the goodness of an algorithm by measuring the quality of the discovered patterns against a set of known ground-truth patterns. We thus use a set of synthetic datasets where the patterns were artificially embedded. We also use some real-word text collections, where documents are labelled by their topic. In this case, the ground truth labels are compared against the transactions groups induced by the given pattern set. Finally, in order to qualitatively evaluate the usefulness of the discovered patterns, we exploit PaNDa$^+$ to detect overlapping communities in a bipartite network (authors vs. publication venues) derived from the DBLP[1] dataset. The results show

• *C. Lucchese and R. Perego are with ISTI-CNR, Pisa I-56123, Italy. E-mail: {claudio.lucchese, raffaele.perego}@isti.cnr.it.*
• *S. Orlando is with DAIS, Università Ca' Foscari, Venice 30172, Italy. E-mail: orlando@unive.it.*

*Manuscript received 22 Mar. 2013; revised 8 Nov. 2013; accepted 12 Nov. 2013. Date of publication 11 Dec. 2013; date of current version 29 Oct. 2014.*
*Recommended for acceptance by J. Bailey.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier 10.1109/TKDE.2013.181*

1. http://www.informatik.uni-trier.de/~ley/db/

that PANDA$^+$ is able to discover high-quality patterns in both synthetic and real-world datasets.

In summary, this paper extends the work presented in [3], [6], and contains several original contributions:

- a unifying formulation of several cost functions that are used by state-of-the-art algorithms to drive their greedy heuristic strategies and to evaluate the quality of the mined patterns;
- a new algorithm, named PANDA$^+$, that can deal with a variety of cost functions, and that is capable to deal with error noise tolerance thresholds [5], thus improving the accuracy of each mined pattern;
- PANDA$^+$ improves over [4], since it can directly optimize the MDL encoding cost proposed therein;
- we discuss a quantitative evaluation of PANDA$^+$ on both synthetic and real-world datasets, where the ground truth patterns are known;
- we show an application of PANDA$^+$ on community detection in bipartite graphs.

The rest of the paper is organized as follows. We first formalize the approximate top-$k$ pattern mining problem and survey state-of-the-art algorithms (Section 2). Then we discuss the framework of the PANDA$^+$ algorithm (Section 3). Finally, we report on the conducted experiments (Section 4), and we discuss other related works (Section 5) and concluding remarks (Section 6).

## 2 Problem Statement and Algorithms

We first introduce some notation useful to state our problem. A *transactional dataset* of $N$ transactions and $M$ items can be represented by a *binary* matrix $\mathcal{D} \in \{0,1\}^{N\times M}$ where $\mathcal{D}(i,j) = 1$ if the $i$–th item occurs in the $j$–th transaction, and $\mathcal{D}(i,j) = 0$ otherwise. An *approximate pattern* $P$ identifies two sets of items/transactions, and is denoted by a pair of binary vectors $P = \langle P_I, P_T\rangle$, where $P_I \in \{0,1\}^M$ and $P_T \in \{0,1\}^N$. The outer product $P_T \cdot P_I^{\mathsf{T}} \in \{0,1\}^{N\times M}$ of the two binary vectors identifies a sub-matrix of $\mathcal{D}$. Being $P$ approximate, it may cover some elements $\mathcal{D}(i,j) = 0$ (*false positives*). Indeed, we say that a set of patterns $\Pi = \{P_1, \ldots, P_{|\Pi|}\}$ approximately covers dataset $\mathcal{D}$, where some occurrences $\mathcal{D}(i,j) = 1$ may not be covered by any pattern in $\Pi$ (*false negatives*). We can finally define a *noise matrix* $\mathcal{N} \in \{0,1\}^{N\times M}$ as follows:

$$\mathcal{N} = \bigvee_{P\in\Pi} (P_T \cdot P_I^{\mathsf{T}}) \quad \veebar \mathcal{D}. \tag{1}$$

where $\vee$ and $\veebar$ are respectively the element-wise *logical or* and *xor* operators. Indeed, if an occurrence $\mathcal{D}(i,j)$ corresponds to either a *false positives* or *false negatives* with respect to $\Pi$, we have that $\mathcal{N}(i,j) = 1$. We can now state our problem as an optimization one.

**Problem 1 (Approximate Top-$k$ Pattern Discovery).** *Given a binary dataset $\mathcal{D} \in \{0,1\}^{N\times M}$ and an integer k, find the pattern set $\overline{\Pi}_k$, $|\overline{\Pi}_k| \leq k$, that minimizes a* cost function *$J(\Pi_k, \mathcal{D})$:*

$$\overline{\Pi}_k = \operatorname*{argmin}_{\Pi_k} J(\Pi_k, \mathcal{D}). \tag{2}$$

In the following we review some cost functions and the algorithms that adopt them. Since the problem belongs to the NP class, these algorithms try to optimize specific functions $J$ with some greedy strategy. In addition, they exploit some specific *parameters*, whose purpose is to make the pattern set $\overline{\Pi}_k$ subject to particular *constraints*, with the aim of (1) reducing the algorithm search space or (2) possibly avoiding that the greedy generation of patterns brings to local minima. As an example of the former type of parameters, we mention the frequency of the pattern. Whereas, for the latter type of parameters, an example is the amount of false positives we can tolerate in each pattern.

### 2.1 Minimizing Noise (ASSO)

ASSO [1] is a greedy algorithm aimed at finding the pattern set $\Pi_k$ that minimizes the amount of noise in describing the input data matrix $\mathcal{D}$. This is measured as the $L^1$-norm $\|\mathcal{N}\|$ (or Hamming norm), which simply counts the number of 1 bits in matrix $\mathcal{N}$ as defined in Eq.(1). ASSO is thus a greedy algorithm minimizing the following function:

$$J_A(\Pi_k, \mathcal{D}) = \|\mathcal{N}\|. \tag{3}$$

Indeed, ASSO aims at finding a solution for the *Boolean matrix decomposition problem*, thus identifying two low-dimensional factor binary matrices of rank $k$, such that their *Boolean product* approximates $\mathcal{D}$. The authors of ASSO named this matrix decomposition problem the Discrete Basis Problem (DBP). It can be shown that the DBP problem is equivalent to the approximate top-$k$ patterns mining problem when optimizing $J_A$ (see [3] for further details). The authors prove that the decision version of the problem is NP-complete by reduction to the set basis problem, and that $J_A$ cannot be approximated within any factor in polynomial time, unless P = NP.

ASSO works as follows. First it creates a set of candidate item sets. For each item pair $(i,j)$, ASSO measures the confidence of the association rule $i \Rightarrow j$. At the end it generates a candidate by combining $i$ with all $j$'s, such that $conf(i \Rightarrow j) \geq \tau$, where the threshold $\tau$ is a parameter of the algorithm. Once determined the candidates, ASSO iteratively selects a pattern from the candidate set by greedily minimizing $J_A$. ASSO was proved to perform better than other matrix decomposition approaches, such as principal component analysis and non-negative matrix factorization, since these last methods were not specifically tailored for binary matrices.

### 2.2 Minimizing the Pattern Set Complexity (HYPER+)

The HYPER+ [2] algorithm works in two phases. In the first phase (corresponding to the covering version HYPER [7]) the algorithm greedily selects a set of patterns $\Pi'$ by minimizing the following cost function that models the pattern set complexity:

$$J_H(\Pi', \mathcal{D}) = \sum_{P\in\Pi'} (\|P_I\| + \|P_T\|). \tag{4}$$

Indeed, the algorithm tries to cover all the 1 bits of $\mathcal{D}$, without false negatives and positives, and thus without any noise. The rationale is to promote the simplest description of the input data $\mathcal{D}$. Note that the size of $\Pi'$ depends on the amount of patterns that suffice to cover the 1-bits in $\mathcal{D}$. A minimum support parameter $\sigma$ is used to select an initial set of (frequent) patterns for starting the greedy selection

TABLE 1
Objective Functions for the Top-*k* Patterns Discovery Problem

| cost function | description |
|---|---|
| $J_A(\Pi_k,\mathcal{D}) = J^+(\Pi_k,\mathcal{D},\rho=0,\gamma_P(P)=0,\gamma_\mathcal{N}(\mathcal{N})=\|\mathcal{N}\|)$ | Minimize noise [1]. |
| $J_H(\Pi_k,\mathcal{D}) = J^+(\Pi_k,\mathcal{D},\rho=1,\gamma_P(P)=\|P_T\|+\|P_I\|,\gamma_\mathcal{N}(\mathcal{N})=0)$ | Minimize pattern set complexity [7]. |
| $J_P(\Pi_k,\mathcal{D}) = J^+(\Pi_k,\mathcal{D},\rho=1,\gamma_P=\|P_T\|+\|P_I\|,\gamma_\mathcal{N}(\mathcal{N})=\|\mathcal{N}\|)$ | Minimize noise and pattern set complexity [6], [3]. |
| $J_P^{\overline{\rho}}(\Pi_k,\mathcal{D}) = J^+(\Pi_k,\mathcal{D},\rho=\overline{\rho},\gamma_P(P)=\|P_T\|+\|P_I\|,\gamma_\mathcal{N}(\mathcal{N})=\|\mathcal{N}\|)$ | Extend $J_P$ to leverage the trade-off between noise and pattern set complexity. |
| $J_E(\Pi,\mathcal{D}) = J^+(\Pi,\mathcal{D},\rho=1,\gamma_P(P)=\mathsf{enc}(P),\gamma_\mathcal{N}(\mathcal{N})=\mathsf{enc}(\mathcal{N}))$ | Minimize the encoding length [8] of the pattern model according to [4]. |

phase, thus reducing the search space of the optimization strategy.

Concerning the second phase of HYPER+, pairs of patterns in $\Pi'$ are recursively merged as long as a new collection $\Pi''$, with a reduced number of patterns, is obtained. In $\Pi''$ some *false positives* can occur, but their number must not exceed a given budget $\beta$. Finally, since $\Pi''$ is ordered (from most to least important pattern), we can simply select $\Pi_k$ as the top-listed $k$ patterns in $\Pi''$, as done by the algorithm's authors in Section 7.4 of [2]. Note that this also introduces *false negatives*, corresponding to all the occurrences $\mathcal{D}(i,j)=1$ in the dataset that remain uncovered after selecting the top-$k$ patterns $\Pi_k$ only.

### 2.3 Minimizing the MDL Encoding

In [4] the MDL principle [8] is adopted to evaluate a pattern set $\Pi_k$. According to the MDL principle, the regularities in $\mathcal{D}$, corresponding to the discovered patterns $\Pi_k$, can be used to *lossless compress* $\mathcal{D}$: thus the best pattern set $\Pi_k$ is the one that induces the smallest encoding of $\mathcal{D}$. More formally, given a collection of pattern sets, the best $\Pi_k$ is the one that minimizes the following cost function:

$$J_E(\Pi_k,\mathcal{D}) = \mathsf{enc}(\Pi_k)+\mathsf{enc}(\mathcal{D}|\Pi_k) = = \sum_{P\in\Pi_k}\mathsf{enc}(P)+\mathsf{enc}(\mathcal{N}) \tag{5}$$

where $\mathsf{enc}(\Pi_k)$ is the length, in bits, of the description of $\Pi_k$, and $\mathsf{enc}(\mathcal{D}|\Pi_k)$ is the length, in bits, of the description of the data when encoded with $\Pi_k$. Optimal codes are assumed. Note that $\mathsf{enc}(\Pi_k)$ is computed in terms of the encoding costs of all the single patterns, whereas $\mathsf{enc}(\mathcal{D}|\Pi_k)$, i.e., the residual information we derive $\mathcal{D}$ when $\Pi_k$ is known, which exactly corresponds to encoding $\mathcal{N}$. Hereinafter, we refer to $J_E$ as the Typed-XOR cost described in [4]. Such cost function encodes independently false negatives and false positives. Authors do not provide an algorithm for mining directly the patterns that minimize $J_E$. Rather, they use $J_E$ to select only the best top-listed patterns returned by ASSO. We show that PANDA [3] can be easily extended to directly optimize the MDL cost $J_E(\Pi_k,\mathcal{D})$, rather than performing a post-pruning as done in [4].

### 2.4 Towards a Generalized Algorithm Framework

In [3], [6] we presented the PANDA algorithm, which minimizes the cost function $J_P$ which is a combination of $J_A$ and $J_H$:

$$J_P(\Pi_k,\mathcal{D}) = J_H(\Pi_k,\mathcal{D})+J_A(\Pi_k,\mathcal{D}) = = \sum_{P\in\Pi_k}(\|P_T\|+\|P_I\|) \ + \ \|\mathcal{N}\| \tag{6}$$

The cost function of PANDA is indeed inspired by the MDL principle. By looking at Eq. (6) we can in fact recognize the cost of the model (the cost of the pattern set $J_H$) and the cost of the dataset given the model (the cost of the noise matrix $J_A$).

In this paper we discuss a greedy algorithmic framework, called PANDA$^+$, which extends PANDA and is able to exploit a generalized cost function in order to deal in a flexible way with a wider class of optimization problems, including the MDL cost $J_E$ discussed above. Moreover, PANDA$^+$ permits the user to define noise constraints over the patterns extracted.

## 3 PANDA$^+$ ALGORITHMIC FRAMEWORK

We exploit a two-stage heuristics to greedily select approximate patterns from the huge solution space of Problem 1 ($2^{M+N}$ candidate patterns). The problem of discovering each pattern is decomposed into two simpler ones: *(i)* discover a noise-less pattern that covers the yet uncovered 1-bits of $\mathcal{D}$, and *(ii)* extend it to form a good approximate pattern, thus allowing some false positives to occur within the pattern. Rather than considering all the possible $2^M$ combinations of items, these are sorted to maximize the probability of generating large cores, and processed one at a time without backtracking.

### 3.1 Cost Functions Generalization

The cost function adopted by PANDA$^+$ to solve Problem 1 is a generalization/extension of the one used in [3], [6] that also includes all of the aforementioned cost functions. Let $J^+$ be this new generalized function:

$$J^+(\Pi_k,\mathcal{D},\rho,\gamma_P,\gamma_\mathcal{N}) = \rho\cdot\sum_{P\in\Pi_k}\gamma_P(P) \ + \ \gamma_\mathcal{N}(\mathcal{N}) \tag{7}$$

where $\mathcal{N}$ is the noise matrix defined by Eq. (1), $\gamma_P$ and $\gamma_\mathcal{N}$ are user defined functions measuring the costs of patterns description and noise, respectively, and $\rho \geq 0$ weights the relative importance of the patterns cost.

Table 1 shows how the cost function defined by Eq. (7) can be instantiated to obtain all the functions discussed above, and allows for new functions to be introduced, by fully leveraging the trade-off between patterns description cost and noise cost (thanks to parameter $\rho$). Note the $J_P^{\overline{\rho}}$ is a generalization of the function $J_P$ already proposed for PANDA in [6] and [3], with parameter $\overline{\rho}$ that determines a different trade-off between patterns description cost and noise cost.

**Algorithm 1** PANDA$^+$

```
Parameters:  $K$ : max no. of patterns to be extracted
             $\mathcal{D}$ : input dataset
             $\mathcal{D}_R$ : residual dataset
             $J$ : instantiation of function $J^+$ (Table 1)
             $C$ : core pattern to extend
             $E$ : items extension list
             $\epsilon_r$ : max row noise threshold
             $\epsilon_c$ : max column noise threshold
             $\Pi$ : set of patterns
1:  PANDA$^+$ ($K$, $\mathcal{D}$, $J$, $\epsilon_r$, $\epsilon_c$)
2:      $\Pi \leftarrow \emptyset$                         ▷ the current collection of patterns
3:      $\mathcal{D}_R \leftarrow \mathcal{D}$             ▷ the residual data yet to be explained
4:      for $iter \leftarrow 1, \ldots, K$ do
5:          $C, E \leftarrow$ FIND-CORE($\mathcal{D}_R$, $\Pi$, $\mathcal{D}$, $J$)
6:          $C^+ \leftarrow$ EXTEND-CORE($C$, $E$, $\Pi$, $\mathcal{D}$, $J$, $\epsilon_r$, $\epsilon_c$)
7:          if $J(\Pi, \mathcal{D}) < J(\Pi \cup C^+, \mathcal{D})$ then
8:              break                                  ▷ $J$ cannot be improved any more
9:          $\Pi \leftarrow \Pi \cup C^+$
10:         $\mathcal{D}_R(i,j) \leftarrow 0 \;\; \forall i,j$ where $C_T^+(i)=1 \wedge C_I^+(j)=1$
11:     return $\Pi$
```

As a result, PANDA$^+$ is the first approximate top-$k$ pattern mining algorithm directly optimizing the MDL-based cost function $J_E$.

### 3.2 Noise Threshold

The original PANDA algorithm may extend a pattern, by adding an item or a transaction, even if some of noise is introduced, provided that its MDL-like cost function improves. This behavior may lead the algorithm to fall into a local minimum. In order to avoid the greedy search strategy accepting too noisy patterns, we introduce into PANDA$^+$ two maximum noise thresholds $\epsilon_r, \epsilon_c \in [0,1]$, inspired by [5], aimed at upper-bounding the maximum amount of noise. Given $P = \langle P_I, P_T \rangle \in \Pi_k$, the following constraints must hold:

1) every item $j$ (*column*) of $P$ (s.t. $P_I(j) = 1$) must be included in at least $(1 - \epsilon_c) \cdot \|P_T\|$ transactions of $P$:

$$\forall j \text{ s.t. } P_I(j) = 1, \sum_{i=1}^{N} (P_T(i) \cdot \mathcal{D}(i,j)) \;\geq\; (1 - \epsilon_c) \cdot \|P_T\|$$

2) every transaction $i$ (*row*) of $P$ (s.t. $P_T(i) = 1$) must include at least $(1 - \epsilon_r) \cdot \|P_I\|$ items of $P$:

$$\forall i \text{ s.t. } P_T(i) = 1, \sum_{j=1}^{M} (P_I(j) \cdot \mathcal{D}(i,j)) \;\geq\; (1 - \epsilon_r) \cdot \|P_I\|$$

### 3.3 Algorithm Overview

PANDA$^+$ adopts a greedy strategy that extracts an ordered sequence of patterns, by progressively increasing the degree of coverage of $\mathcal{D}$. As stated above, we decompose the problem of pattern detection into two simpler problems: discovering of a noise-less pattern, and its extension by allowing false positives to occur in the pattern. For the former sub-problem, similarly to AC-CLOSE [5], we assume that, even in presence of noise, within an approximate pattern $P$ occurring in $\mathcal{D}$ we can identify a smaller *core pattern*, made up of *true positive* bits only. That is, given a pattern $P = \langle P_I, P_T \rangle$, there exists a smaller subset pattern $C = \langle C_I, C_T \rangle$ such that:

**Algorithm 2** Function FIND-CORE()

```
1:  function FIND-CORE($\mathcal{D}_R$, $\Pi$, $\mathcal{D}$, $J$)
2:      $E \leftarrow \emptyset$                                   ▷ item extension list
3:      $S = \{s_1, \ldots, s_M\} \leftarrow$ SORT-ITEMS-IN-DB($\mathcal{D}_R$)
4:      $C \leftarrow \left\langle C_T = 0^N, C_I = 0^M \right\rangle$
5:      $C_I(s_1) \leftarrow 1$
6:      $C_T(i) \leftarrow 1 \quad \forall i$ where $\mathcal{D}_R(i, s_1) = 1$
7:      for $h \leftarrow 2, \ldots, M$ do
8:          $C^* \leftarrow C$                                  ▷ create a new candidate
9:          $C_I^*(s_h) \leftarrow 1$
10:         $C_T^*(i) \leftarrow 0 \quad \forall i$ where $\mathcal{D}_R(i, s_h) = 0$
11:         if $J(\Pi \cup C^*, \mathcal{D}) \leq J(\Pi \cup C, \mathcal{D})$ then
12:             $C \leftarrow C^*$
13:         else
14:             $E$.append($s_h$)
15:     return $C, E$
```

$$\begin{array}{rl} (i) & C_T(i) = 1 \Longrightarrow P_T(i) = 1; \\ (ii) & C_I(j) = 1 \Longrightarrow P_I(j) = 1; \\ (iii) & (C_T(i) = 1 \wedge C_I(j) = 1) \Longrightarrow \mathcal{D}(i,j) = 1. \end{array}$$

For the latter sub-problem, given a core pattern $C$, PANDA considers all the items and transactions to further enlarge $C$, with the aim of detecting a better approximate pattern, until the largest extension of $C$ that minimizes the overall cost $J^+$ is found. In order to reduce complexity, the items to add are picked according to a given sort order.

The pseudo code of Algorithm 1 gives an overview of PANDA$^+$. It iterates the two main steps at most $K$ times, where $K$ is the input parameter that determines the maximum number of patterns to be extracted. Note that during each iteration, first a core pattern $C$ is extracted (line 5), which is then extended to form a new approximate pattern $C^+$ (line 6). If $C^+$ reduces the current cost of the model, it is added to the pattern set $\Pi$. Regardless of the user-provided parameter $K$, PANDA stops generating further patterns when the one under examination does not improve the cost of the model (line 7).

In order to discover new core patterns (line 10), PANDA$^+$ uses a particular view $\mathcal{D}_R$ of $\mathcal{D}$, called *residual dataset*. We have that $\mathcal{D}_R(i,j) = 1$ *iff* $\nexists P \in \Pi$ such that $P_T(i) = 1$ and $P_I(j) = 1$. The rationale is that we want to start discovering a new pattern from portions of $\mathcal{D}$ not yet covered by any previous pattern.

### 3.4 Extraction of Dense Cores

The function that extracts dense cores is shown in Algorithm 2. The extension list $E$ is initially empty.

The items in $\mathcal{D}_R$ are selected in a specific sort order (line 3) and evaluated for being added to a core pattern without backtracking. This heuristic technique reduces the search space explored, but also favors the discovery of large patterns when a proper item ordering is used. We discuss a number of possible orderings in Section 3.6.

A core pattern is initialized with the first item $s_1$ in the ordered list $S$, and its supporting transactions in $\mathcal{D}_R$. The remaining items in $S$ are processed one by one. The current item $s_h$ is used to create a new candidate pattern $C^*$ (lines 8-10). Since we do not allow false positives when we add items to a core pattern (line 9), we can have a reduction in the number of supporting transactions (line 10). If $C^*$ reduces the cost of the pattern set with respect to the

**Algorithm 3** Function EXTEND-CORE()

```
1: function EXTEND-CORE($C$, $E$, $\Pi$, $\mathcal{D}$, $J$, $\epsilon_r$, $\epsilon_c$)
2:     $added\_item \leftarrow True$
3:     while $added\_item$ do
                                                  ▷ add new transactions
4:         for $i \in \{1, \dots, N\}$ where $C_T(i) = 0$ do
5:             $C^* \leftarrow C$;   $C^*_T(i) \leftarrow 1$
6:             if NOT_TOO_NOISY($C^*, \epsilon_r, \epsilon_c$) then
7:                 if $J(\Pi \cup C^*, \mathcal{D}) \leq J(\Pi \cup C, \mathcal{D})$ then
8:                     $C \leftarrow C^*$
9:         $added\_item \leftarrow False$
10:        while $E \neq \emptyset$ do                ▷ add a single new item
11:            $e \leftarrow E$.pop()
12:            $C^* \leftarrow C$;   $C^*_I(e) \leftarrow 1$
13:            if NOT_TOO_NOISY($C^*, \epsilon_r, \epsilon_c$) then
14:                if $J(\Pi \cup C^*, \mathcal{D}) \leq J(\Pi \cup C, \mathcal{D})$ then
15:                    $C \leftarrow C^*$
16:                    $added\_item \leftarrow True$
17:                    break
18:    return $C$                                  ▷ return the extended core $C$
```

current core $C$, then $C^*$ is promoted to be the new candidate (line 12) and it is used in the subsequent iteration. Otherwise, $s_h$ is appended to the extension list $E$ (line 14), and it can be used later to extend the extracted dense core. Eventually, every item occurring in $\mathcal{D}_R$ is processed only once: it is either added to the dense core $C$, or appended to the extension list $E$. The procedure returns $C$ and $E$ for further processing. To speed up the computation, our implementation exploits a vertical representation of $\mathcal{D}$ by storing per-item transaction id lists (a.k.a. tid-lists).

The complexity of Algorithm 2 depends on the cost of the initial sorting of items, which is $O(M \log(M))$, plus the cost of extracting the new candidate core, which consists in scanning the tid-lists of size $M$ associated with the $N$ items $s_h$ in $S$. Since the model cost induced by the new core can be computed in a constant time, this second step has complexity $O(NM)$. Furthermore, since the number of transactions $N$ is likely to be larger than $\log(M)$, we can state the following.

**Proposition 1.** *The computational complexity of Algorithm 2 is $O(MN)$.*

## 3.5 Extension of Dense Cores

Given a dense core $C = \langle C_I, C_T \rangle$, the procedure shown in Algorithm 3 iteratively tries to add transactions to $C_T$ and items to $C_I$, possibly introducing some false positives, as long as the cost function $J$ is reduced.

The first step extends the current $C$ with additional transactions, even if they do not include all of its items. A transaction $t_i$, not yet included in $C_T$, is used to create a new candidate pattern $C^*$ (line 5). If this transaction extension does not introduce too much false positives with respect to both error parameters $\epsilon_r$ and $\epsilon_c$ (line 6), and the new pattern carries an improvement over the previous one (line 7), then $C^*$ is promoted and considered for further extensions in place of C. This step ends when all the transactions have been considered.

In the second step, items from the extension list $E \subset \mathcal{I}$ are considered. A single item $e$ is removed from the head of list $E$, and added to $C_I$, thus forming a new pattern $C^*$ (line 12). If this item extension introduces too much false positives with respect to both error parameters $\epsilon_r$ and $\epsilon_c$ (line 13), or this new pattern does not improve the overall cost function (line 14), then item $e$ is disregarded. This step is repeated till a new item can be profitably added to $C$ (lines 15–17), or list $E$ becomes empty.

Once a new item is added to $C$, this extension cycle is repeated. The two-step cycle stops when $E$ becomes empty.

Note that when an item $e$ is added to pattern $C$, the corresponding transaction set $C_T$ is not modified: therefore, we assume that $e$ is approximately supported by all the transactions in $C_T$, and we allow some of them, up to a threshold $\epsilon_c$, to not support the added item $e$ (false positives). We behave similarly when we extend $C_T$ by adding transaction $i$: we do not modify $C_I$, and thus assume that all the items in $C_I$ are approximately supported by transaction $i$, and we allow some of them, up to a threshold $\epsilon_r$, to not be supported by the added transaction $i$ (false positives).

There are some interesting subtleties in the evaluating process of new pattern $C^*$. First $C^*$ may introduce some false positives, thus affecting $\mathcal{N}$. At the same time, the extension may cover some occurrences of items that were already considered noise due to some previously extracted pattern, and this noisy bits do not have to be accounted again in the cost function $\gamma_{\mathcal{N}}$ of Equation (7). If the balance between the bits covered and the noise introduced justifies the increased representation cost of $C^*$, then the new pattern $C^*$ is accepted.

Second, the inclusion of $C^*$ in $\Pi$ depends on the amount of covered bits that were not already covered by other patterns in $\Pi$. For example, suppose that $C^*$ does not introduce any noise, and that all the bits covered thanks to the extension were already covered by another pattern in $\Pi$. Then, the only variation in the overall cost function is due to the cost of representing $\Pi$, which increases when we add items/transactions to a pattern, thus making the extended pattern $C^*$ not convenient.

Regarding the cost for Algorithm 3, the first and the second step are repeated at most $M$ times, once for each checked item. In order to improve the efficiency of the extension operations an histogram is used that stores the number of pattern's items contained in each transaction of the dataset, and records already covered occurrences, and two other histograms to store the number of noisy occurrences per row and per column.

In the first step, up to $N$ transactions are added. The cost of the new candidate and the per-transaction maximum noise can be computed in constant time by exploiting the aforementioned histograms. The maximum noise thresholds require to also check the introduced noise per item, with a total cost of $O(MN)$.

In the second step, adding a new item requires a single tid-list intersection with a cost $O(N)$. Per-item noise is given by the result of the intersection, while the per-transaction noise threshold is checked in $O(N)$.

**Proposition 2.** *The computational complexity of Algorithm 3 is $O(M(MN + N)) = O(M^2N)$.*

## 3.6 Scoring and Sorting Items

The item ordering used by procedure FIND-CORE (Algorithm 2, line 3) affects the whole algorithm behavior. Indeed, items have to be associated with a *score*, and then ordered by descending score. Such ordering is a greedy choice aiming at discovering the largest patterns first. We consider three scoring strategies, and the possible impact on the algorithm complexity:

- **Frequency.** The score of an item is given by its frequency in $\mathcal{D}_R$.
- **Couples frequency.** As in [9], the score of an item $j$ is given by the sum of the supports of every pair $(j, h)$ occurring in the transactions of $\mathcal{D}_R$.
- **Correlation.** The score of item $i$ is given by the support of $\hat{C}_I \cup i$, where $\hat{C}_I$ is the item set corresponding to the binary vector $C_I$ of the *current* candidate pattern.

The rationale of the *frequency* strategy is that the most frequent items has to have the highest probability of being part of a core pattern. The other two strategies aim at grouping together items that, to some extent, are highly correlated.

The *frequency* scoring is the least expensive one to compute, since it requires a single database scan with cost $O(Nm)$, where $m = \|\mathcal{D}\|/N$ is the average transaction length in the worst case. This does not affect Proposition 1.

The *couples frequency* strategy requires to accumulate the count of each item pair during the scan of the dataset, thus increasing the complexity of the scoring step to $O(N \cdot m^2)$. Still, this does not affect Proposition 1 as long as $m \leq \sqrt{M}$.

The *correlation* ordering has a stronger impact, since every time we extend $C$, i.e., after line 12 of Algorithm 2, we have to afresh recompute the correlation score of each item before sorting. Item correlation thus requires a scan of the dataset to be repeated at most $M$ times, for an additional cost of $O(M \cdot Nm)$. This cost overcomes what stated by Proposition 1, and if $m$ is large the cost of Algorithm 2 may become nearly quadratic in the number of items $M$.

### 3.7 Randomized PaNDa$^+$ and Complexity

We introduce randomization in PaNDa in order to reduce the risk of being trapped in local minima. The lines 5 and 6 of Algorithm 1 are repeated for $R$ randomization rounds. During each round, the ordering of items is perturbed, and each item is thus associated with a *randomized score*. The amount of randomization allowed changes at each round. Once $R$ candidate patterns are found, only the best is evaluated for its inclusion in $\Pi$.

Regarding the overall complexity of our algorithm, PaNDa$^+$ simply invokes Algorithm 2 and Algorithm 3, and then builds $\mathcal{D}_R$ for each of the $K$ (or less) patterns to be extracted. The cost of constructing $\mathcal{D}_R$ can be also bounded by $O(MN)$. Furthermore, if randomization is included, this cost is increased by a factor of $R$.

**Proposition 3.** *The computational complexity of the* PaNDa$^+$ *Algorithm is* $O(RKM^2N)$, *where* $R$ *is the number of randomization rounds,* $K$ *is the maximum number of patterns to be extracted,* $M$ *and* $N$ *are respectively the number of items and transactions in the dataset* $\mathcal{D}$.

## 4 Experiments

Quantitatively evaluating the goodness of a set of patterns discovered by a top-$k$ mining tool may be difficult and controversial. This is particularly true when the actual patterns that are present in a given dataset are unknown, and therefore there is no *ground truth* that we can use as a benchmark.

We performed this evaluation by exploiting two sets of datasets, synthetic and on real-world ones. In the former set, patterns were artificially embedded into the data, and thus a underlying ground truth is known. Regarding the latter set of experiments, we instead used the class labels of some publicly available collections of text classification documents as indicators of patterns occurrences.

In addition, in order to show the usefulness of the discovered patterns in a real setting, we applied PaNDa$^+$ to a bipartite network (authors vs. publication venues) derived from the DBLP dataset[2].

TABLE 2
Parameters of the Synthetic Dataset Generator

| Parameter | Meaning | Value |
|---|---|---|
| $N$ | Number of transactions | 10,000 |
| $M$ | Number of items | 100 |
| $K$ | Number of embedded patterns | $\{5, 10, 15, 20\}$ |
| $min_I$ | Min no. of items in a pattern | 3 |
| $max_I$ | Max no. of items in a pattern | 15 |
| $min_T$ | Min no. of transactions in a pattern | 500 |
| $max_T$ | Max no. of transactions in a pattern | 5000 |
| $n$ | Bit flipping probability | $\{0\%, 3\%, 5\%, 7\%\}$ |
| $\omega$ | Maximum pattern overlap ratio | 0.50 |

### 4.1 Synthetic Dataset Generation

For the first group of experiments we create a number of synthetic datasets. We embed a given known set of patterns $\Omega$ – i.e., our *ground truth* – into an *empty* dataset, and then introduce noise by randomly flipping a given percentage of bits in the corresponding binary representation. Since the ground-truth patterns $\Omega$ embedded in each dataset are known, it is thus possible to compare them with the patterns extracted by any mining algorithm.

The parameters of our synthetic dataset generator are the number of transactions $N$ and items $M$, the number of embedded patterns $K = |\Omega|$, and the two pairs $(min_I, max_I)$ and $(min_T, max_T)$. A pattern $P \in \Omega$ is generated by selecting uniformly at random a set of items $P_I$, $min_I \leq \|P_I\| \leq max_I$, and a set of transactions $P_T$, $min_T \leq \|P_T\| \leq max_T$. Some additional parameters are used to control the amount of noise introduced and the maximum overlap between the embedded patterns in order to avoid to have multiple patterns explaining the same portion of the dataset. In particular, we add noise to the matrix by flipping values from 0 to 1, and vice-versa, with uniform probabilities $n^+$ and $n^-$, respectively. When $n^+ = n^-$, we denote by $n$ the overall noise level. Moreover, given a maximum overlap ratio $\omega$, the generation of $\Omega$ guarantees that for any two embedded patterns $P^1, P^2 \in \Omega$, we have that $P_I^1 \cdot P_I^2 \leq \omega \cdot \min(\|P_I^1\|, \|P_I^2\|)$, and $P_T^1 \cdot P_T^2 \leq \omega \cdot \min(\|P_T^1\|, \|P_T^2\|)$. Table 2 illustrates the parameters used to generate our datasets.

2. The source code of the implementation is available at the authors' website http://goo.gl/3ITl5z

TABLE 3
Parameter Sweeping Settings

| Algorithm | Parameter | Values |
|---|---|---|
| ASSO | $\tau$ | $[0.05, 1.00]$ with steps of $0.05$ |
| HYPER+ | $\beta$ | $1\%, 5\%, 10\%$ |
| | $\sigma$ | $5\%, 10\%, 20\%, 50\%, 80\%$ |
| PANDA$^+$ | $\epsilon_r$,$\epsilon_c$ | $[0.0, 1.0]$ with steps of $0.1$ |
| | items order | frequency, couples frequency, correlation |
| | $R$ | 20, no randomization |

## 4.2 MDL-Based Cost Optimization and Model Order Selection in Synthetic Datasets

The authors of [4] proposed to exploit the MDL principle to estimate the number of patterns present in a given dataset. First, they used ASSO to extract a ordered list of $k$ patterns. Then, they selected the first $\hat{k}$ patterns of the list such that the cost $J_E(\Pi_{\hat{k}}, \mathcal{D})$ of encoding the dataset is minimized. We use exactly the same procedure to select the best patterns extracted by ASSO, HYPER+ and PANDA$^+$, and we base our considerations on the selected patterns only. Note that when evaluating the ASSO patterns, we indeed compare against the approach of [4]. Of course, this post-mining selection does not apply to PANDA$^+$ $J_E$ which does not produce patterns that do not improve the $J_E$ cost function.

We first investigate the following issues: (a) which algorithm is the best at optimizing the $J_E$ cost, and (b) which algorithm is the best at guessing the number of patterns embedded in the dataset.

Since ASSO, HYPER+, and PANDA$^+$ need a number of parameters to be tuned, similarly to [4] we run parameter sweeping to find the set of parameters leading to the pattern set with the smallest cost $J_E$.

In our experiments, we used the best performing variant of the ASSO algorithm which is named ASSO + *iter* by varying its single parameter $\tau$. For what regards HYPER+, we fine-tuned the minimum support threshold $\sigma$ and noise budget $\beta$ parameter as done in Section 7.4 of [2]. Finally, we tested independently four variants of PANDA$^+$ optimizing different cost functions: $J_A$, $J_P$, $J_P^{1.5}$, $J_E$. For $J_P$ we also fixed $\epsilon_r = \epsilon_c = 1$, so as to mimic the original PANDA algorithm [3]. Table 3 reports the different parameters values tested in our experiments.

For every algorithm, we asked for extracting $k = 64$ patterns, as this is the maximum value allowed by the original implementation of ASSO. Note that each algorithm may generate $k' < k$ patterns when they are not able to optimize further their own cost function. Of each generated pattern set, we considered only the first extracted $\hat{k}$ patterns leading the smallest encoding cost $J_E$.

Fig. 1 shows how the selection procedure based on MDL encoding affects the different algorithms when mining the artificial dataset with 20 patterns. When $n = 0\%$, the selection procedure has no effect. In Fig. 1 we can see that the normalized encoding cost $J_E(\Pi_{\hat{k}}, \mathcal{D})/J_E(\emptyset, \mathcal{D})$, $\bar{J}_E$ for short, of each algorithm monotonically decreases when the number of extracted patterns increases. Note that only PANDA$^+$ $J_P^{1.5}$ and PANDA$^+$ $J_E$ are able to detect the correct number of patterns. The other algorithms are not able to handle well overlapping patterns, and they need to create more than 20 patterns to cover the full dataset. When $n = 7\%$, the MDL principle has a significant effect. If we consider the smallest $J_E$ cost achieved, which is highlighted with filled marks in the figures, we can observe that ASSO generates 64 patterns, but only the first 20 improve the $J_E$ cost, thus matching the correct number of patterns embedded in the data. Indeed, the ASSO algorithm just aims at covering the data, and therefore it is likely to produce overfitted patterns to cover also noisy occurrences. We have that $\hat{k} = 20$ also for PANDA$^+$ $J_P$, and a similar behavior is observed for $J_A$ and PANDA$^+$ $J_P^{1.5}$. It is interesting to note that the cost function $J_P^{1.5}$ can lead to results that are similar to the ones of the MDL encoding $J_E$. The HYPER+ algorithm does not show good performance.

We note that ASSO never resulted to be the best algorithm in optimizing the MDL encoding cost. Therefore, any conclusion based on the synergy of ASSO algorithm and MDL principle should be carefully weighted. Finally, The effectiveness of PANDA$^+$ $J_E$ in finding pattern sets with small costs suggests a greater robustness.

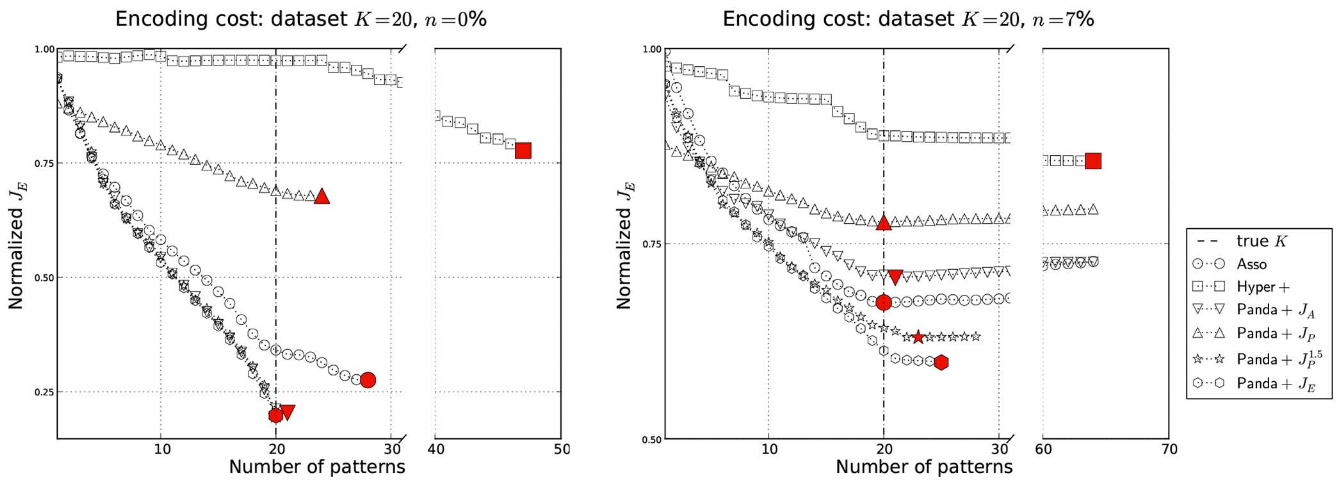


Fig. 1. Normalized $J_E$ cost as a function of the number of patterns extracted. The best $\hat{k}$ is highlighted.

TABLE 4
Encoding Cost Minimization on Synthetic Datasets

| dataset | | ASSO | | HYPER+ | | PANDA$^+$ $J_A$ | | PANDA$^+$ $J_P$ | | PANDA$^+$ $J_P^{1.5}$ | | PANDA$^+$ $J_E$ | | PANDA$^+$ $J_E$ vs. ASSO |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $K$ | $n$ | $\hat{k}$ ($k'$) | $\overline{J}_E$ | $\hat{k}$ ($k'$) | $\overline{J}_E$ | $\hat{k}$ ($k'$) | $\overline{J}_E$ | $\hat{k}$ ($k'$) | $\overline{J}_E$ | $\hat{k}$ ($k'$) | $\overline{J}_E$ | $\hat{k}$ ($k'$) | $\overline{J}_E$ | $\Delta\overline{J}_E$ |
| 5 | 0% | *5* (5) | **0.10** | 9 (9) | 0.13 | *5* (5) | **0.10** | *5* (5) | **0.10** | *5* (5) | **0.10** | *5* (5) | **0.10** | +0% |
| 5 | 3% | *5* (64) | **0.39** | 62 (64) | 0.55 | *5* (64) | **0.39** | *5* (64) | **0.39** | *5* (6) | **0.39** | *5* (5) | **0.39** | +0% |
| 5 | 5% | *5* (64) | **0.51** | 57 (64) | 0.60 | *5* (64) | **0.51** | *5* (64) | **0.51** | *5* (13) | **0.51** | *5* (5) | **0.51** | +0% |
| 5 | 7% | *5* (64) | 0.61 | 59 (64) | 0.70 | *5* (64) | **0.60** | *5* (64) | **0.60** | *5* (23) | **0.60** | *5* (5) | **0.60** | -1% |
| 10 | 0% | *10* (10) | **0.13** | 32 (32) | 0.17 | *10* (10) | **0.13** | *10* (10) | 0.22 | *10* (10) | **0.13** | *10* (10) | **0.13** | +0% |
| 10 | 3% | *10* (61) | **0.36** | 64 (64) | 0.64 | *10* (64) | 0.37 | *10* (64) | 0.44 | *10* (39) | **0.36** | *10* (10) | **0.36** | +0% |
| 10 | 5% | 11 (64) | 0.51 | 64 (64) | 0.73 | *10* (64) | 0.49 | *10* (64) | 0.53 | *10* (34) | 0.48 | 11 (11) | **0.47** | -7% |
| 10 | 7% | *10* (64) | 0.63 | 64 (64) | 0.76 | *10* (64) | 0.57 | *10* (64) | 0.60 | *10* (33) | **0.56** | *10* (10) | **0.56** | -11% |
| 15 | 0% | *15* (15) | **0.16** | 56 (60) | 0.46 | 16 (16) | 0.17 | *15* (15) | 0.57 | *15* (15) | **0.16** | *15* (15) | **0.16** | +0% |
| 15 | 3% | 20 (64) | 0.39 | 64 (64) | 0.73 | 17 (64) | 0.43 | *15* (64) | 0.61 | 20 (23) | 0.38 | 16 (16) | **0.36** | -9% |
| 15 | 5% | 17 (64) | 0.52 | 64 (64) | 0.77 | 17 (64) | 0.53 | *15* (64) | 0.67 | 18 (18) | 0.48 | 16 (16) | **0.46** | -12% |
| 15 | 7% | 17 (64) | 0.61 | 64 (64) | 0.82 | 16 (64) | 0.63 | *15* (64) | 0.72 | 17 (36) | 0.57 | 17 (17) | **0.55** | -11% |
| 20 | 0% | 28 (28) | 0.28 | 47 (47) | 0.78 | 21 (21) | **0.20** | 24 (24) | 0.68 | *20* (20) | **0.20** | *20* (20) | **0.20** | -28% |
| 20 | 3% | 23 (64) | 0.47 | 64 (64) | 0.82 | 28 (64) | 0.51 | 22 (64) | 0.68 | 23 (24) | 0.43 | 23 (23) | **0.41** | -14% |
| 20 | 5% | 24 (64) | 0.59 | 64 (64) | 0.87 | 21 (64) | 0.61 | 21 (64) | 0.73 | 24 (24) | 0.54 | 23 (23) | **0.50** | -16% |
| 20 | 7% | *20* (64) | 0.67 | 64 (64) | 0.86 | 21 (64) | 0.71 | *20* (64) | 0.78 | 23 (28) | 0.63 | 25 (25) | **0.60** | -11% |

Table 4 reports, for each test dataset and noise level, the values of $\hat{k}$, $k'$ and $\overline{J}_E$. HYPER+ is never able to guess the correct number of patterns, and it is the worst performing algorithm in terms of $J_E$ optimization. The datasets having only 5 embedded patterns are easily mined by ASSO and by all the variants of PANDA$^+$, as they correctly guess the number of patterns and achieve about the same encoding cost. PANDA$^+$ $J_E$ is clearly the best performing algorithm dominating all the others in terms of $J_E$ optimization. We observe that PANDA$^+$ $J_E$ always uses $R = 20$ randomization rounds and the correlation items ordering to achieve the best encoding cost, but there is not an overall best setting for $\epsilon_r$ and $\epsilon_c$.

The problem of discovering the correct number of patterns deserves a careful discussion. The two algorithms PANDA$^+$ $J_E$ and ASSO exhibit similar performance with a small advantage of the former. The fact that PANDA$^+$ $J_E$ is also better at optimizing $J_E$ suggests that it is the best choice among the two. On the other hand, PANDA$^+$ $J_P$ is able to guess the right number of patterns in 17 tests out of 20, but it never provides an encoding cost better than PANDA$^+$ $J_E$. In conclusion, we observe that the $J_E$ cost function is a useful stopping criterion for a greedy top-$k$ mining algorithm, but it is not possible to draw any conclusive remark on its capability to detect the correct number of patterns in the data.

## 4.3 Measures of Pattern Quality

In this section we present the measures used to evaluate the quality of the pattern set ($\Pi$) extracted by the various algorithms with respect to a ground-truth ($\Omega$). We adopt three measures $F^I(\Pi,\Omega)$, $F^T(\Pi,\Omega)$, and $F^P(\Pi,\Omega)$. They are inspired by the well-known F-measure [10] usually employed in *clustering* evaluation.

The overall F-measure, where clusters and classes may overlap, is computed as a weighted mean of the various F-measures for all the classes of the ground-truth, where the F-measure for a class $j$ is computed by selecting the best matching cluster $i$, i.e., $\max_i F(i,j)$.

Keeping this into consideration, it should be easier to understand the rationale of our three measures $F^I(\Pi,\Omega)$, $F^T(\Pi,\Omega)$, and $F^P(\Pi,\Omega)$, where the patterns in $\Pi$ play the roles of the clusters detected by an algorithm, and those in $\Omega$ of the classes of the ground-truth.

We first introduce the F-measures over the items (columns) of the detected patterns, namely $F^I(\Pi,\Omega)$. Given patterns $P \in \Pi$ and $Q \in \Omega$, we define the item-based precision and recall as:

$$p^I(P,Q) = \frac{\|P_I \wedge Q_I\|}{\|P_I\|} \qquad r^I(P,Q) = \frac{\|P_I \wedge Q_I\|}{\|Q_I\|}$$

The weighted item-based F-measure $F^I(\Pi,\Omega)$ is thus computed as follow:

$$F^I(\Pi,\Omega) = \frac{\sum_{Q\in\Omega} \|Q_I\| \cdot \max_{P\in\Pi} \frac{2\cdot p^I(P,Q)\cdot r^I(P,Q)}{p^I(P,Q)+r^I(P,Q)}}{\sum_{Q\in\Omega} \|Q_I\|}$$

$F^T(\Pi,\Omega)$ is the F-measure over the transactions (rows) of the detected patterns. It is analogous to $F^I(\Pi,\Omega)$, where we use $P_T$ and $Q_T$ for each pattern $P \in \Pi$ and $Q \in \Omega$, respectively.

The last measure $F^P(\Pi,\Omega)$ refers to the whole patterns rather than items and transactions only. Given two patterns $P \in \Pi$ and $Q \in \Omega$, precision and recall are computed by considering the sets of 1 bits in the matrices $P_T \cdot P_I^{\mathsf{T}}$ and $Q_T \cdot Q_I^{\mathsf{T}}$.

Note that computing $F^P(\Pi,\Omega)$ is stronger than a mere *reconstruction error* of the ground truth. For example, a pattern $Q \in \Omega$ in the ground truth could be reconstructed correctly by overlapping several extracted patterns, but a simple reconstruction error measure is not able to penalize such misbehavior. On the other hand, $F^P(\Pi,\Omega)$ measures to what extent, for each $Q \in \Omega$, there is a pattern $P \in \Pi$ that matches $Q$. Finally, note that when $F^P(\Pi,\Omega)$ is equal to 1, we also have a perfect reconstruction of the ground truth.

For the sake of completeness, we also use the *significance* of a pattern set, as proposed by Gupta *et al.* [11], denoted by $\chi(\Pi,\Omega)$. However, it only considers the matching items (columns) in the patterns.

## 4.4 Patterns Quality in Synthetic Datasets

Table 5 reports the patterns quality observed according to the $\chi$, $F^I$ and $F^P$ measures. These new experiments confirm the previous results. The HYPER+ algorithm achieves significantly worse results than competitors. PANDA$^+$ $J_P$ performs generally worse than ASSO. Recall that PANDA$^+$ $J_P$ is the only variant of PANDA$^+$ presented here that does

TABLE 5
Patterns Quality on Synthetic Datasets

| dataset | | ASSO | | | HYPER+ | | | PANDA$^+$ $J_A$ | | | PANDA$^+$ $J_P$ | | | PANDA$^+$ $J_P^{1.5}$ | | | PANDA$^+$ $J_E$ | | | PANDA$^+$ $J_E$ vs. ASSO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $K$ | $n$ | $\chi$ | $F^I$ | $F^P$ | $\chi$ | $F^I$ | $F^P$ | $\chi$ | $F^I$ | $F^P$ | $\chi$ | $F^I$ | $F^P$ | $\chi$ | $F^I$ | $F^P$ | $\chi$ | $F^I$ | $F^P$ | $\Delta\chi$ | $\Delta F^I$ | $\Delta F^P$ |
| 5 | 0% | **1.00** | **1.00** | **1.00** | 0.92 | **1.00** | 0.63 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | +0% | +0% | +0% |
| 5 | 3% | **1.00** | **1.00** | **1.00** | 0.78 | 0.71 | 0.71 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | +0% | +0% | +0% |
| 5 | 5% | **1.00** | **1.00** | **1.00** | 0.87 | 0.85 | 0.63 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | +0% | +0% | +0% |
| 5 | 7% | 0.97 | 0.97 | 0.97 | 0.89 | 0.85 | 0.55 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | +3% | +3% | +2% |
| 10 | 0% | **1.00** | **1.00** | **1.00** | 0.89 | **1.00** | 0.59 | **1.00** | **1.00** | **1.00** | 0.96 | 0.97 | 0.95 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | +0% | +0% | +0% |
| 10 | 3% | **1.00** | **1.00** | **1.00** | 0.73 | 0.64 | 0.53 | **1.00** | **1.00** | 0.99 | 0.94 | 0.95 | 0.92 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | +0% | +0% | +0% |
| 10 | 5% | 0.93 | 0.93 | 0.95 | 0.68 | 0.55 | 0.49 | **1.00** | **1.00** | 0.98 | 0.95 | 0.95 | 0.92 | **1.00** | **1.00** | 0.99 | **1.00** | **1.00** | **1.00** | +7% | +7% | +5% |
| 10 | 7% | 0.90 | 0.89 | 0.90 | 0.79 | 0.69 | 0.38 | **1.00** | **1.00** | 0.97 | 0.96 | 0.96 | 0.93 | **1.00** | **1.00** | **0.99** | **1.00** | **1.00** | **0.99** | +11% | +13% | +10% |
| 15 | 0% | **1.00** | **1.00** | **1.00** | 0.87 | 0.93 | 0.54 | **1.00** | **1.00** | 0.99 | 0.92 | 0.94 | 0.78 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | +0% | +0% | +0% |
| 15 | 3% | 0.90 | 0.91 | 0.92 | 0.38 | 0.36 | 0.34 | 0.99 | 0.98 | 0.93 | 0.92 | 0.94 | 0.87 | 0.99 | **1.00** | 0.97 | **1.00** | **1.00** | **0.98** | +10% | +10% | +6% |
| 15 | 5% | 0.81 | 0.82 | 0.83 | 0.74 | 0.63 | 0.34 | 0.97 | 0.97 | 0.91 | 0.92 | 0.94 | 0.85 | 0.99 | 0.99 | 0.95 | **1.00** | **1.00** | **0.98** | +24% | +21% | +18% |
| 15 | 7% | 0.78 | 0.78 | 0.80 | 0.75 | 0.62 | 0.36 | 0.93 | 0.92 | 0.83 | 0.90 | 0.92 | 0.71 | **0.99** | **1.00** | 0.93 | **0.99** | 0.99 | **0.96** | +27% | +27% | +20% |
| 20 | 0% | 0.97 | 0.96 | 0.94 | 0.43 | 0.36 | 0.33 | **1.00** | **1.00** | 0.99 | 0.89 | 0.92 | 0.73 | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | **1.00** | +3% | +4% | +7% |
| 20 | 3% | 0.78 | 0.79 | 0.77 | 0.44 | 0.34 | 0.28 | 0.96 | 0.96 | 0.88 | 0.90 | 0.92 | 0.71 | **0.99** | 0.98 | 0.93 | **0.99** | **0.99** | **0.95** | +28% | +26% | +23% |
| 20 | 5% | 0.74 | 0.75 | 0.76 | 0.40 | 0.32 | 0.26 | 0.95 | 0.94 | 0.82 | 0.84 | 0.86 | 0.54 | 0.98 | 0.98 | 0.90 | **0.99** | **0.99** | **0.95** | +35% | +33% | +25% |
| 20 | 7% | 0.73 | 0.71 | 0.73 | 0.73 | 0.62 | 0.32 | 0.88 | 0.88 | 0.73 | 0.87 | 0.88 | 0.63 | **0.97** | **0.97** | 0.85 | 0.96 | 0.96 | **0.91** | +31% | +35% | +25% |

TABLE 6
Pattern Quality on Text Datasets

| | ASSO | | | | | PANDA$^+$ $J_P^{1.5}$ | | | | | PANDA$^+$ $J_E$ | | | | | PANDA$^+$ $J_E$ vs. ASSO | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| dataset | $F^T$ | $\overline{J}_E$ | $\hat{k}$ | $Len.$ | $Supp.$ | $F^T$ | $\overline{J}_E$ | $\hat{k}$ | $Len.$ | $Supp.$ | $F^T$ | $\overline{J}_E$ | $\hat{k}$ | $Len.$ | $Supp.$ | $\Delta F^T$ | $\Delta\overline{J}_E$ |
| R8 | 0.367 | 0.888 | 61 | 7.05 | 432.4 | 0.593 | 0.866 | 64 | 6.66 | 245.7 | **0.652** | **0.852** | 64 | 6.97 | 473.8 | +78% | -4% |
| R52 | 0.475 | 0.902 | 64 | 1.30 | 1435.2 | 0.514 | 0.884 | 64 | 6.64 | 273.5 | **0.536** | **0.866** | 55 | 11.82 | 536.1 | +13% | -4% |
| 20NG | 0.123 | 0.907 | 13 | 6.46 | 4727.3 | 0.125 | 0.888 | 17 | 9.88 | 1426.9 | **0.143** | **0.875** | 7 | 33.86 | 3468.4 | +16% | -3% |
| WebKB | 0.411 | 0.941 | 14 | 12.21 | 514.4 | 0.517 | 0.914 | 63 | 6.57 | 203.9 | **0.546** | **0.906** | 14 | 24.71 | 534.7 | +33% | -4% |

not exploit the noise thresholds $\epsilon_r$ and $\epsilon_c$. However, the capability of directly optimizing $J_A$ allows PANDA$^+$ $J_A$ to outperform ASSO in most cases. Consider, for instance, the most complex dataset $K = 20$ and $n = 7\%$, where PANDA$^+$ $J_A$ scores $\chi = .88$ compared with ASSO with $\chi = .73$. Interestingly, an improvement is observable for PANDA$^+$ $J_P^{1.5}$. This means that the weighting of the patterns cost and the introduction of the noise thresholds $\epsilon_r$ and $\epsilon_c$ allow a better robustness to noise.

The overall best performing algorithm is PANDA$^+$ $J_E$. If we restrict our analysis to the most severe measure $F_P$, which requires to correctly detect both the items and the supporting transactions of a pattern, then PANDA$^+$ $J_E$ always achieves the best results. In particular, if we consider again the dataset with $K = 20$ and $n = 7\%$ the improvement of PANDA$^+$ $J_E$ over ASSO, according to the three measures $\chi$, $F^I$, and $F_P$, is of 31%, 35% and 25%, respectively.

On the basis of the experiments we conducted on synthetic datasets, we can conclude that PANDA$^+$ $J_E$ is the best algorithm among those we considered not only at minimizing the cost function $J_E$, since it is optimized directly, but also, and more importantly, in discovering the true patterns embedded in the datasets, even in presence of noise.

## 4.5 Patterns Quality on Real-World Text Datasets

The second group of experiments was conducted on a set of four text datasets typically used for label categorization: R8 and R52 (from Reuters 21578), 20NG and WebKB[3]. Table 7 reports some characteristics of these datasets. Each document is labeled according to the topic discussed. In this context, a good pattern is formed by a set of terms frequently occurring in documents discussing the same topic, thus identifying the topic itself. Approximate patterns are very relevant since a topic can be easily identified by a set of distinguishing terms, which, however, do not usually occur in *every* document about the same topic. Even if the set $\Omega$ of ground truth patterns is not known, since the distinctive terms of each topic are not known, it is possible to exploit the class label information of a document to measure the goodness of a pattern.

We used $F^T(\Pi, \Omega)$ to estimate the goodness of the extracted patterns. The ground-truth $\Omega$ is derived from $\mathcal{C} = \{C_1, C_2, \ldots,\}$, where $C_i \in \mathcal{C}$ is the set of transactions belonging to the $i$-th topic. Thus each $Q^T$ associated with $Q \in \Omega$ is a binary vector representation of a distinct class $C_i \in \mathcal{C}$. Note that $Q^I$ is not used to evaluate the quality of the patterns extracted, since we do not have any supervised knowledge about which are the best terms that describes a class of documents.

We remark that the goal of this experiment is not to show that PANDA$^+$ improves over the state of the art of pattern-based classification algorithms. To this end, PANDA$^+$ or ASSO should be deeply re-designed to take into account class labels, and embedded into a more complex framework such as [12]. Our objective is to use $F^T$ as a proxy of the patterns quality, since a large value of $F^T$ means that some of the discovered patterns have high correlation with the dataset labels.

Before extracting patterns from the input datasets, we removed the class label information, which was used only during the evaluation. In addition, as usual with text dataset, we disregarded irrelevant terms by removing those occurring in more the 90% and less than 1% of the documents, i.e., rare and stop words. For the datasets 20NG and WebKB, the minimum frequency threshold was raised to 5% because ASSO was not able to complete successfully the parameter sweeping process. The resulting dataset were transformed into a transactional format by mapping

3. http://web.ist.utl.pt/~acardoso/datasets/

TABLE 7
Text Datasets Statistics

| dataset | classes | items | transactions | avg. tr. length |
|---|---|---|---|---|
| R8 | 8 | 17387 | 7674 | 40.08 |
| R52 | 52 | 19241 | 9100 | 42.15 |
| 20NG | 20 | 70216 | 18821 | 83.83 |
| WebKB | 4 | 7770 | 4199 | 77.22 |

the terms of each document to numerical identifiers. We used parameter sweeping for both ASSO and PANDA$^+$ as with synthetic dataset, and we selected the run achieving the smallest cost for the best number of patterns $\hat{k}$. Therefore, the choice of the algorithms' parameters is guided by the cost function $J_E$ and entirely unsupervised. Even in this case, $R = 20$ randomization rounds and correlation items ordering is the best performing setting for PANDA$^+$.

The resulting patterns were evaluated by computing their transaction-based F-measure $F^T$, which is reported in Table 6. In every experiment, PANDA$^+$ $J_E$ and PANDA$^+$ $J_P^{1.5}$ outperform ASSO, especially on the R8 dataset. PANDA$^+$ $J_E$ dominates the other two. A reduction in the normalized encoding cost as small as 4% can lead to an improvement of 78% in the F-measure. An interesting difference of PANDA$^+$ $J_E$ with respect to the other two algorithms is given by the size of the extracted patterns, which are consistently larger in the number of items. This is expected as the $J_E$ cost function demotes false negatives more than false positives, and therefore is more tolerant of noisy patterns.

## 4.6 Patterns Extracted from a Real Social Network

In this section we discuss the results of using PANDA$^+$ as a tool to identify communities from bipartite graphs extracted from a real-wold dataset. To this end, we mined a dump of the DBLP database, spanning a period that goes from January 2008 to July 2013. We built a bipartite graph of *authors* and *publication venues*, where an edge links an author $a$ to a publication venue $v$ *iff* $a$ published at least a paper at conference/journal $v$ regardless the year of publication. We discarded authors who published in less than 5 venues, and venues that hosted less than 40 distinct authors. We also removed the venue "Computing Research Repository" (CoRR), which is a free online multidisciplinary repository and it is not useful to our purpose of detecting topical communities. From this bipartite graph, we generated the incidence binary matrix $\mathcal{D}$, where rows are authors and columns are venues. The resulting binary matrix $\mathcal{D}$ includes $N = 112{,}261$ authors, and $M = 4{,}283$ venues.

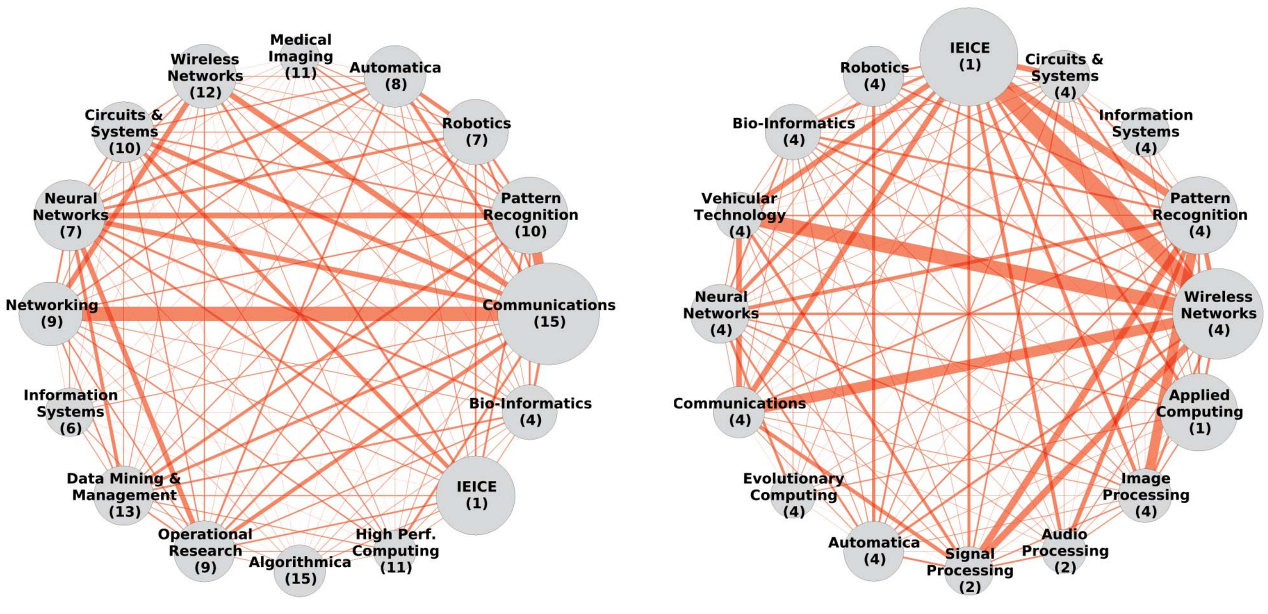
We run PANDA$^+$ $J_E$ (with $R = 20$ and correlation item ordering) by varying the two maximum noise thresholds $\epsilon_r$ and $\epsilon_c$. After testing several combinations, we observed that all of them are very useful for studying, inspecting and understanding the diverse overlapping social communities present in the bipartite DBLP graph. Small noise thresholds generate small communities covering a very specific topic. By increasing the noise thresholds, it is possible to analyze how the communities merge together into wider topics. Due to space constraints, we only discuss two parameter configurations: (i) $\epsilon_r = 1.0$, $\epsilon_c = 1.0$, and (ii) $\epsilon_r = 0.5$, $\epsilon_c = 0.8$. In the case *(i)* we do not give any constraint over the amount of noise contained in each pattern, so that the noise is only limited by cost function $J_E$. In the case *(ii)* instead, any extracted pattern should contain authors that published in at least 50% of the pattern's venues, and venues that contain at least 20% of the pattern's authors. The setting *(ii)* of the two parameters is reasonable, considering that there are many more authors than venues. The algorithms ASSO and HYPER+ were not able to complete within one hour and therefore they are not discussed here. This is because their implementation does not handle efficently large and sparse data.

It is worth remarking that DBLP does not always disambiguate homonymous authors. As a result, PANDA$^+$ extracted a pattern with more than 50 venues and as little as 200 authors. By manual inspection, we found that the venues of the pattern covered very different topics, and that authors has common Chinese surnames (see Li, Liu, Wang, Yang, Zhang, etc.). This pattern was clearly due to the presence of homonymous authors, and we thus did not consider it in the following discussion.

Fig. 2 shows the top-16 patterns extracted by PANDA$^+$ $J_E$. Each pattern is represented by a node, whose label was manually chosen according to the venues composing it. The size of the node is proportional to the number of authors contained, whereas the number of venues is shown between parentheses. The sharing of authors among patterns is instead captured by the thickness of the edges. Finally, nodes are placed according to their extraction order, the first being the rightmost and then proceeding counter-clock wise.

When using the larger noise thresholds of case *(i)*, see Fig. 2(a), the extracted patterns are larger. *Communications* is the largest pattern, and it has a significant sharing of authors with *Networking*. Despite the strong sharing of authors, there is only one overlapping venue (ICC). By inspecting the patterns' venues in Table 8, the reader can appreciate that although the topics of the two patterns are related, they are separated enough: the former is concerned with low level signal transmissions, whereas the latter mainly regards network protocols. Other strong links are present between *Operational Research* and *Neural Networks* and between *Pattern Recognition* and *Communications*. Interestingly enough, the IEEE TKDE journal falls into the *Data Mining & Management* pattern. The cluster of venues appears to be very reasonable, including some information retrieval conferences (see WSDM). Also, it contains the most influential authors in the Data Mining field according to Microsoft Academic Search: Jiawei Han, Philip S. Yu, Rakesh Agrawal, Christos Faloutsos, Hans-Peter Kriegel, Eamonn J. Keogh, George Karypis, and Heikki Mannila. Finally, readers of this journal may appreciate the quality of the venue groups *Algorithmica* and *High Perf. Computing*.

Fig. 2(b) shows the top-16 patterns extracted with the less tolerant configuration of case *(ii)*. Due to space constraints, we do not illustrate in detail their venue composition. We observe, that the extracted patterns have a finer granularity. The previous *Communications* related clusters are now separated in more focused groups: *Wireless Networks*, which includes ICC, VTC



(a) (b)

Fig. 2. Top-16 patterns mined by PANDA$^+$ $J_E$ from the DBLP database with noise thresholds $\epsilon_r$ and $\epsilon_c$. (a) $\epsilon_r$ = 1.0, $\epsilon_c$ = 1.0. (b) $\epsilon_r$ = 0.5, $\epsilon_c$ = 0.8.

Spring, GLOBECOM, and WCNC, *Communications*, made up of ICC, GLOBECOM, IEEE Communications Letters, and IEEE Transactions on Communications, *Vehicular Technology*, composed of CC, PIMRC, IEEE Transactions on Vehicular Technology and IEEE Transactions on Wireless Communications, and, finally, *Signal Processing*, which is composed of ICASSP and IEEE Transactions on Signal Processing.

Another interesting example is given by the *Pattern Recognition* cluster, which is now separated from the 3 related sub-fields of *Signal Processing*, *Audio Processing* and *Image Processing*, each of them having a large author overlap with the new more focused *Pattern Recognition* pattern. Note is also well connected to *Wireless Networks*.

Finally, note that in both Fig. 2(a) and (b) there is a pattern labeled by *IEICE*, composed of a single venue, namely IEICE Transactions, which contains an abnormal number of authors (about 6,000). It is a publication of an Asiatic organization which mainly contains papers on electronics and communications.

## 5 Related Work

We classify related works in three large categories: matrix decomposition based, database tiling, Minimum Description Length and frequent item sets based. All of them have many similarities with the Top-*k* Pattern Discovery Problem. However, there are four features that are considered altogether only by our framework: (a) our model is specifically tailored for binary datasets, (b) our model allows for overlapping patterns, (c) our model minimizes the error with respect to the original dataset *and* the encoding cost of the pattern set discovered, (d) frequent item sets need not to be extracted from the original dataset. In the following, we illustrate in more detail the algorithms falling in the aforementioned categories.

TABLE 8
Some Groups of Publication Venues Referring to Fig. 2(a)

| Topic (#Venues) | Group of Publication Venues | No. of Authors |
|---|---|---|
| Communication (15) | ICC, IEEE Transactions on Signal Processing, VTC Fall, VTC Spring, IEICE Transactions, EURASIP J. Adv. Sig. Proc., ICASSP, IEEE T. Vehicular Technology, WCNC, PIMRC, IEEE Transactions on Wireless Communications, GLOBECOM, IEEE Communications Letters, EURASIP J. Wireless Comm. and Networking, IEEE Transactions on Communications | 6373 |
| Wireless Networks (12) | EURASIP J. Wireless Comm. and Networking, IEEE Transactions on Communications, VTC Fall, VTC Spring, IEEE Communications Letters, WCNC, Wireless Personal Communications, PIMRC, IEEE Transactions on Wireless Communications, GLOBECOM, Wireless Sensor Network, IEEE T. Vehicular Technology | 2524 |
| Networking (9) | ICC, IEEE Trans. Parallel Distrib. Syst., IEEE Journal on Selected Areas in Communications, Computer Networks, Computer Communications, Int. Journal of Network Management, IEEE Trans. Mob. Comput., IEEE/ACM Trans. Netw., INFOCOM | 2509 |
| Data Mining & Management (13) | CIKM, AAAI, EDBT, SIGIR, ICDE, SIGMOD Conference, WSDM, ICDM, MLDM, WWW (Companion Volume), PVLDB, IEEE Trans. Knowl. Data Eng., KDD | 2178 |
| Algorithmica (15) | Electronic Colloquium on Computational Complexity (ECCC), Inf. Process. Lett., ISAAC, Theory Comput. Syst., FOCS, STACS, ESA, Theor. Comput. Sci., APPROX-RANDOM, STOC, SODA, ACM Transactions on Algorithms, SIAM J. Comput., ICALP (1), Algorithmica | 1644 |
| High Perf. Computing (11) | IEEE Trans. Computers, IPDPS, IPDPS Workshops, MASCOTS, IEEE Trans. Parallel Distrib. Syst., SC, J. Parallel Distrib. Comput., ICPP, CCGRID, ICPADS, Concurrency and Computation: Practice and Experience | 1106 |

**Matrix decomposition based.** The methods in this class aim at finding a product of matrices that describes the input data with a smallest possible amount of error. Probabilistic latent semantic indexing (PLSI) [13] is a well known technique that solves the above decomposition problem. PLSI was initially devised to model co-occurrence of terms in a corpus of documents $\mathcal{D}$. However, PLSI is not formulated for binary inputs, and thus multiple occurrences of the same term can be modelled in the real-value matrix. The core of PLSI is a generative model called *aspect model*, where each database transaction results from the contribution of every class variable, in turn associated with a subset of items. The model is obtained by computing the probabilities of classes/items for each transaction/item. Similar approaches for non binary inputs have been studied, such as Latent Dirichlet allocation (LDA) [14], Independent Component Analysis (ICA) [15], Non-negative Matrix Factorization, and others [17]. However, evaluations studies suggest that these models cannot be trivially adapted to binary datasets [16].

An improvement over these models is proposed in [18] and [1]. In [18] the authors discuss a different generative model, where an item occurs if it is generated by at least one class variable. The proposed algorithm, called LIFT, is able to discover patterns when dominant items are present, i.e. items that are generated with high probability by one class variable only. Finally, we already discussed the Discrete Basis Problem [1], and the proposed greedy algorithm ASSO, which makes use of items' correlation statistics. This happens to be a limitation in many cases, since *global* statistics may be too general to catch local correlations. Both [18] and [1] aim at minimizing the noise matrix $\mathcal{N}$ only.

**Database tiling.** Database tiling algorithms are tailored for the discovery of large patterns in *binary* datasets. They differ on the notion of pattern adopted.

The maximum $K$-tiling problem introduced in [19] requires to find the set of $K$ tiles, *noise-less* and possibly overlapping, having the largest coverage of the given database $\mathcal{D}$. However, this approach does not handle the false positives present in the data. Co-clustering [20] is a borderline approach between tiling and matrix decomposition. It is formulated as a matrix decomposition problem, and therefore its objective is to approximate the input data by minimizing $\mathcal{N}$.

According to [21], tiles can be hierarchical. A basic tile is indeed a hyper-rectangle with density $p$. A tile might contain several non overlapping sub-tiles, i.e., exceptional regions with larger or smaller density. Unlike our approach, low-density regions are considered as important as high density ones, and inclusion of tiles is preferred instead of overlapping. The MDL principle is used to chose the best tile set.

**Minimum Description Length principle.** According to [22], [23], a set of item sets is interesting if and only if it yields a good lossless compression of the database. A set of item sets, called *cover* or *code table*, is used to encode all the transactions in the database, meaning that every transaction is represented by the union of some item sets in the cover. The MDL principle is used to choose the best code table. There are two significant differences from our approach. First, patterns that cover a given transaction must be disjoint, and this increases the total number of extracted patterns, which becomes more specialized. Second, false positives are not allowed, and noise is handled by disregarding transactions being hard to encode. However, the process of building a code table requires an expensive selection process. In their experiments, the authors exploited the collection of frequent item sets, mined with a very low minimum support (even a single transaction) to achieve good results.

The Bayesian Information Criterion (BIC), which has a theoretical support through the MDL principle, can also be used to score the goodness of a set of patterns. In [24], BIC is used by the greedy algorithm MTV to find a small number of patterns that fit well the input data. MTV starts from the knowledge of the frequency of singleton items, and, at each iteration, heuristically adds to the current solution the item set whose frequency is most surprising according to the current maximum entropy model. The algorithm stops when it is not possible to improve the BIC score of the current pattern set.

In [25] the authors propose a novel framework for the comparison of different pattern sets. From a given pattern set, a probability distribution over $\mathcal{D}$ is derived according to the maximum entropy principle, and then the Kullback-Leibler distance between these two distributions is used to measure the dissimilarity of two pattern sets. Given a set of labelled transactions, this distance is used to *redescribe* a given pattern set with a subset of patterns that best matches the class labels, i.e. the input clusters. This redescription process is applied to several several unsupervised clustering and pattern mining algorithms, including K-MEANS, ASSO, HYPER+, KRIMP, MTV and others. The authors conclude that HYPER+ and ASSO are the two best performing algorithms, i.e., the ones producing the set of patterns whose redescription is closer to the underlying class label distribution. This result supports our choice of HYPER+ and ASSO as competitor algorithms of PANDA$^+$.

Recently, Miettinen [26] proposed a method to dynamically update a Boolean matrix factorization when new data, even new rows/columns, are added to the to the original matrix.

**Frequent item sets.** The classical definition of frequent item set requires that all the items of each mined set actually occur in the supporting transactions. In order to deal with noisy databases, the common approach is to relax the notion of *support* of an item set. *Weak* error tolerant item sets (ETI) [27] are the first example of such relaxation apprach. A pattern $P = \langle P_I, P_T \rangle$ is said to be *weak* ETI *iff* $\|P_T\| \geq \sigma$, and it has at most $\epsilon$ false positives. The previous constraint allows also transactions that do not contain any item of the pattern to be included in $P_T$. *Strong* ETI introduced an error tolerance threshold $\epsilon_r$ to be enforce on any transactions of $P_T$. Unfortunately, the enumeration of all weak/strong ETI requires to explore the full item sets space. Approximate frequent item sets (AFI) [28] are an extension of strong ETI, where a row-wise and a column-wise tolerance thresholds ($\epsilon_r$ and $\epsilon_c$) are enforced, like in PANDA$^+$. In [5] the notion of closed approximate frequent item sets (AC-AFI) is introduced, and the authors also define the *core pattern*, whose notion is exploited in the PANDA$^+$ algorithm. In [11] all of the above frequent item sets algorithms, plus some of their extensions are evaluated over a collection of synthetic datasets.

In conclusion, algorithms to mine frequent item sets require a demanding data processing, and, more importantly, they tend to generate a large and very redundant collection of patterns. The cost model embedded in our framework implicitly limits the pattern explosion, since a large number of patterns does not minimize the representation cost. The cost model may also allow to rank patterns according to the contributed cost reduction.

## 6 CONCLUSION

In this paper we have discussed the problem of mining approximate top-*k* patterns from binary matrices. Our analysis has explored ASSO, HYPER+, and PANDA, three state-of-the-art algorithms that differ for the greedy strategy adopted and the cost function optimized. We have shown that all these cost functions can be unified into a unique formulation and plugged into a flexible algorithmic framework, named PANDA$^+$, that allows to greedily mine approximate patterns according to several cost functions. Moreover, we have added to this framework two other important features: the possibility of directly optimizing the MDL encoding cost of the solution, and the possibility of dealing with noise constraints, in order to improve the greedy heuristics.

Pattern evaluation is a very complex task due to the lack of benchmarks. We created a benchmark with a set of artificial datasets. We evaluated the ability to estimate the number of true patterns in the data, and the ability to discover both the items and the supporting transactions of the true patterns. The accuracy of PANDA$^+$ was very high, even when a large amount of noise was injected.

We also compared the various algorithms on a few text collections, where documents are annotated by their topic. In this case, the true patterns are not known. Nevertheless, document labels induce a partitioning the collection that we compared with the groups of documents induced by the mined approximate patterns. This kind of evaluation cannot be as accurate as with synthetic datasets, but the goodness of PANDA$^+$ with respect to the state-of-the-art algorithms was confirmed.

Finally we qualitatively evaluated PANDA$^+$, by inspecting some very interesting patterns, identifying social communities, extracted by a DBLP-based bipartite network (authors vs. publication venues). In these tests, we were unable to compare our patterns with the ones returned by ASSO and HYPER+, since these two algorithms were not able to handle so large and sparse datasets, and they did not complete within a reasonable time. On the other hand, although the execution time of PANDA$^+$ is quadratic in the number of items, it completed in about 20 minutes, with 20 randomization rounds, i.e., about one minute for each round.

In conclusion, we showed how the presented PANDA$^+$ algorithmic framework includes the most recent advances in the field, and extends them successfully.

## ACKNOWLEDGMENTS

This work was supported in part by the EU-projects InGeoCLOUDS 297300, MIDAS 318786, E-CLOUD 325091, SECURE! FESR PorCreo 2007-2011, and in part by Industria2015 MOTUS MS01_00015.

## REFERENCES

[1] P. Miettinen, T. Mielikainen, A. Gionis, G. Das, and H. Mannila, “The discrete basis problem,” *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 10, pp. 1348–1362, Oct. 2008.
[2] Y. Xiang, R. Jin, D. Fuhry, and F. F. Dragan, “Summarizing transactional databases with overlapped hyperrectangles,” *Data Min. Knowl. Discov.*, vol. 23, no. 2, pp. 215–251, Sep. 2011.
[3] C. Lucchese, S. Orlando, and R. Perego, “Mining top-k patterns from binary datasets in presence of noise,” in *Proc. SDM SIAM*, 2010, pp. 165–176.
[4] P. Miettinen and J. Vreeken, “Model order selection for Boolean matrix factorization,” in *Proc. KDD*, New York, NY, USA, 2011, pp. 51–59.
[5] H. Cheng, P. S. Yu, and J. Han, “AC-Close: Efficiently mining approximate closed itemsets by core pattern recovery,” in *Proc. ICDM*, Hong Kong, China, 2006, pp. 839–844.
[6] C. Lucchese, S. Orlando, and R. Perego, “A generative pattern model for mining binary datasets,” in *Proc. SAC*, Sierre, Switzerland, 2010, pp. 1109–1110.
[7] Y. Xiang, R. Jin, D. Fuhry, and F. F. Dragan, “Succinct summarization of transactional databases: An overlapped hyperrectangle scheme,” in *Proc. KDD*, Las Vegas, NV, USA, 2008, pp. 758–766.
[8] J. Rissanen, “Modeling by shortest data description,” *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.
[9] M. J. Zaki and C.-J. Hsiao, “Efficient algorithms for mining closed itemsets and their lattice structure,” *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 462–478, Apr. 2005.
[10] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Boston, MA, USA: Addison-Wesley, 2006.
[11] R. Gupta, G. Fang, B. Field, M. Steinbach, and V. Kumar, “Quantitative evaluation of approximate frequent pattern mining algorithms,” in *Proc. KDD*, Las Vegas, NV, USA, 2008, pp. 301–309.
[12] W. Fan *et al.*, “Direct mining of discriminative and essential frequent patterns via model-based search tree,” in *Proc. KDD*, Las Vegas, NV, USA, 2008, pp. 230–238.
[13] T. Hofmann, “Probabilistic latent semantic indexing,” in *Proc. SIGIR*, Berkeley, CA, USA, 1999, pp. 50–57.
[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.
[15] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York, NY, USA: John and Wiley, 2001.
[16] H. Hiisila and E. Bingham, “Dependencies between transcription factor binding sites: Comparison between ICA, NMF, PLSA and frequent sets,” in *Proc. ICDM*, Washington, DC, USA, 2004, pp. 114–121.
[17] Z. Y. Zhang, T. Li, C. Ding, X. W. Ren, and X. S. Zhang. “Binary matrix factorization for analyzing gene expression data,” *Data Min. Knowl. Discov.*, vol. 20, no. 1, pp. 28–52, 2010.
[18] J. Seppänen, E. Bingham, and H. Mannila, “A simple algorithm for topic identification in 0-1 data,” in *Proc. PKDD*, Cavtat-Dubrovnik, Croatia, 2003, pp. 423–434.
[19] F. Geerts, B. Goethals, and T. Mielikäinen, “Tiling databases,” in *Proc. Discov. Sci.*, Padova, Italy, pp. 278–289, 2004.
[20] T. Li, “A general model for clustering binary data,” in *Proc. KDD*, Chicago, IL, USA, 2005, pp. 188–197.
[21] A. Gionis, H. Mannila, and J. Seppänen, “Geometric and combinatorial tiles in 0-1 data,” in *Proc. PKDD*, Pisa, Italy, 2004, pp. 173–184.
[22] A. Siebes, J. Vreeken, and M. van Leeuwen, “Item sets that compress,” in *Proc. SDM SIAM*, 2006.
[23] J. Vreeken, M. van Leeuwen, and A. Siebes, “KRIMP: Mining itemsets that compress,” *Data Min. Knowl. Discov.*, vol. 23, no. 1, pp. 169–214, 2011.
[24] S. Hanhijärvi *et al.*, “Tell me something i don’t know: Randomization strategies for iterative data mining,” in *Proc. KDD*, Paris, France, 2009, pp. 379–388.

[25] N. Tatti and J. Vreeken, "Comparing apples and oranges–Measuring differences between data mining results," in *Proc. ECML-PKDD*, Berlin, Germany, 2011, pp. 398–413.
[26] P. Miettinen, "Dynamic Boolean matrix factorizations," in *Proc. ICDM*, Washington, DC, USA, 2012, pp. 519–528.
[27] C. Cheng, U. Fayyad, and P. Bradley, "Efficient discovery of error-tolerant frequent itemsets in high dimensions," in *Proc. KDD*, San Francisco, CA, USA, 2001, pp. 194–203.
[28] M. Steinbach, P. Tan, and V. Kumar, "Support envelopes: A technique for exploring the structure of association patterns," in *Proc. KDD*, New York, NY, USA, 2004, pp. 296–305.

**Claudio Lucchese** (http://hpc.isti.cnr.it/∼claudio) received the Ph.D. degree in computer science from the University of Venice, Venezia, Italy, in 2008. He is a Researcher with the Italian National Research Council, ISTI-CNR, Pisa, Italy. His current research interests include large-scale data mining techniques for information retrieval. He has published over 50 papers on these topics in peer-reviewed international conferences and journals. He has also participated in EU-funded projects, and served as a Program Committee Member in several data mining and information retrieval conferences.

**Salvatore Orlando** (http://www.dais.unive.it/∼orlando) received the M.Sc. and the Ph.D. degrees in computer science from the Università di Pisa, Pisa, Italy, in 1985 and 1991, respectively. He is an Associate Professor with the Università Ca' Foscari, Venice, Italy. His current research interests include data and web mining, information retrieval, and parallel/distributed systems. He has published over 100 papers in journals and conference proceedings. He has co-chaired conferences, conference tracks, and workshops, and served on the program committees of several premier conferences.

**Raffaele Perego** (http://hpc.isti.cnr.it/∼raffaele) is a Senior Researcher with ISTI-CNR, Pisa, Italy, where he leads the HPC Laboratory (http://hpc.isti.cnr.it/). His current research interests include high performance computing, web information retrieval, and data mining. He has co-authored over 100 papers on these topics published in journals and in proceedings of international conferences. He served as a program committee member in several information retrieval conferences, and coordinated activities in many EC-projects.