This is the author's preprint version. The original article appeared in
Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-022-00846-z

The Minimum Description Length Principle for Pattern Mining: A Survey

Esther Galbrun

Received: 10 March 2021 / Accepted: 20 May 2022

Abstract Mining patterns is a core task in data analysis and, beyond issues of efficient enumeration, the selection of patterns constitutes a major challenge. The Minimum Description Length (MDL) principle, a model selection method grounded in information theory, has been applied to pattern mining with the aim to obtain compact high-quality sets of patterns. After giving an outline of relevant concepts from information theory and coding, we review MDL-based methods for mining different kinds of patterns from various types of data. Finally, we open a discussion on some issues regarding these methods.

Keywords data mining, pattern mining, frequent itemset mining, minimum description length principle, information theory

1 Introduction

Our aim is to review the development of pattern mining methods based on and inspired from the Minimum Description Length (MDL) principle. Although this is an unrealistic goal, we strive for completeness in our coverage of these methods.

Mining frequent patterns is a core task in data mining, and itemsets are probably the most elementary and best studied type of pattern. Soon after the introduction of the frequent itemset mining task (Agrawal et al., 1993), it became obvious that beyond issues of efficiency (Agrawal and Srikant, 1994; Mannila et al., 1994), the problem of selecting patterns constituted a major challenge to tackle, lest the analysts drown under the deluge of extracted patterns.

E. Galbrun
School of Computing, University of Eastern Finland
KPY Novapolis G, Microkatu 1
FI-70210, Kuopio, Finland
E-mail: esther.galbrun@uef.fi

Various properties and measures have been introduced to select itemsets (Webb and Vreeken, 2013; Geng and Hamilton, 2006). They include identifying representative itemsets (Bastide et al., 2000), using user-defined constraints to filter itemsets (De Raedt and Zimmermann, 2007; Soulet et al., 2011; Guns et al., 2013), considering dependencies between itemsets (Jaroszewicz and Simovici, 2004; Yan et al., 2005; Tatti and Heikinheimo, 2008; Mampaey, 2010) and trying to evaluate the statistical significance of itemsets (Webb, 2007; Gionis et al., 2007; Tatti, 2010; Hämäläinen and Webb, 2018), also looking into alternative approaches to explore the search space (Boley et al., 2011; Guns et al., 2011). Initially focused on individual itemsets, approaches were later introduced to evaluate itemsets collectively, trying for instance to identify and remove redundancy (Gallo et al., 2007), also in an iterative manner (Hanhijärvi et al., 2009; Boley et al., 2013). The goal hence moved from mining collections of good patterns to mining good collections of patterns, which is also the main objective when applying the MDL principle.

Before we go any further, let us explain more precisely what we consider, for the present purpose, to constitute *patterns*. Patterns are about repetitions. We adopt the point of view that patterns express the repeated presence in the data of particular items, attribute values or other discrete properties. We divide them into two main categories.

Itemsets are strict conjunctive patterns that require the occurrence of all involved items for the pattern to be considered as occurring in a transaction. For a given dataset, it is thus straightforward to determine where an itemset occurs and where it does not, that is, to compute the pattern's support. Vice versa, when it is known that an itemset occurs in a given transaction, no further information is necessary, as it implies that all items must be present. These concepts naturally extend beyond transactional data. However, such occurrence requirements are fairly strict, and it can be useful to consider more relaxed patterns. In particular, one might use disjunctions, allowing patterns to express a choice between involved items or attributes. Given a dataset, one can then still straightforwardly determine where a pattern holds. On the other hand, knowing that the pattern occurs no longer unambiguously provides information about which item or attribute is present. More or less additional information is needed, depending for instance on whether it is an inclusive or exclusive disjunction. We refer to such patterns that express the presence of a specific substructure in the data as *substructure patterns*.

As an alternative way to relax occurrence requirements, patterns might express that selected items or properties are typically present but not all need always occur, for instance by means of density thresholds. In that case, which data instances belong to the support of the pattern, or vice-versa where each item or property holds, needs to be specified explicitly as it is not directly implied. Rather than the occurrence of a specific substructure, it is then the homogeneity of repetitions within the area delimited by the selected instances and attributes that is of interest. Because such patterns can be seen as delineating homogeneous rectangles in the data, we refer to them as *block patterns*.

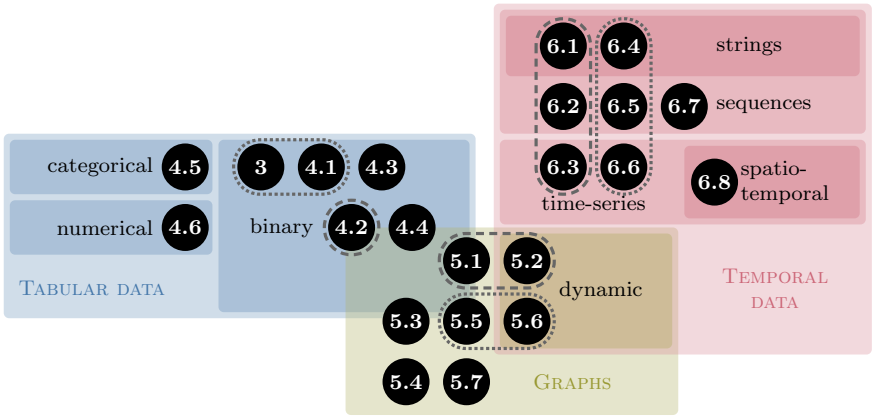
Furthermore, one is typically looking for a collection of patterns and might impose constraints on the overlap between them, that is, require that the patterns involve disjoint sets of attributes, characterise disjoint sets of instances, or both. In particular, the patterns might be required to form a partition of the data, dividing all instances and all attributes into disjoint subsets. Such a partitioning requirement is incompatible with a strict occurrence requirement in practice, in the sense that it is not in general possible to identify a collection of substructure patterns that forms a partition of the data. On the other hand, block patterns might be required to form a partition of the data, corresponding roughly to biclustering approaches, or they might be allowed to overlap, as in tiling approaches.

In summary, we adopt a rather broad definition of what constitute patterns, from itemsets to biclusters over discrete data. However, we stop short of considering clusters more in general as patterns. Clustering constitutes another important field of data mining beside—and partially overlapping with—pattern mining. There, the goal is to organise data instances into groups such that instances within the same group are similar to each other and dissimilar to instances in other groups. Clustering often handles continuous data, typically relying on a concept of distance. Here, we focus on formalisms and methods that are by nature more discrete.

The reader is expected to be familiar with common pattern mining tasks and techniques, but not necessarily with concepts from information theory and coding, of which we therefore give an overview in Section 2, before presenting the two main encoding strategies that use respectively substructure and block patterns. In Sections 3–6, we turn to the review of MDL-based methods for pattern mining proper. The methods are grouped first by the type of data, then by the type of patterns considered, as outlined in Fig. 1. Our focus is on the various encodings designed for these different types of data and patterns, rather than on algorithmic issues related to searching the patterns. In Section 3, we start with one of the thickest branches, stemming from the Krimp algorithm for mining itemsets from transactional data. We continue with itemsets and with other types of patterns for tabular data in Section 4, followed by graphs and temporal data in Sections 5 and 6, respectively. Finally, we consider some discussion points, in Section 7. For further context about the development of the principle and discussion of related problems, please refer to the extended manuscript.[1] In addition, the main characteristics and bibliographic details of publications from Sections 3–6 have been collected into a searchable table.[2]

[1] See `https://arxiv.org/abs/2007.14009`.

[2] See `https://nurblageij.github.io/MDL4PM_survey/`.



3 Mining itemsets with Krimp & Co

4.1 More itemsets
4.2 Blocks in binary data
4.3 Formal Concept Analysis (FCA)
4.4 Boolean Matrix Factorisation (BMF)
4.5 Categorical data
4.6 Numerical data

5.1 Grouping nodes into blocks
5.2 Grouping nodes into blocks in dynamic graphs
5.3 Finding hyberbolic communities
5.4 The Map Equation
5.5 Identifying substructures
5.6 Identifying substructures in dynamic graphs
5.7 Finding pathways between nodes

6.1 Finding haplotype blocks
6.2 Segmenting sequences
6.3 Segmenting timeseries
6.4 Mining substrings
6.5 Mining episodes from sequences
6.6 Mining motifs from timeseries
6.7 Mining periodic patterns
6.8 Trajectories

Fig. 1 Organisation of Sections 3–6. MDL-based methods for pattern mining are grouped first by the type of data (tabular, graph or temporal), then by the type of patterns and strategies considered (blocks or substructures). Numbers refer to sections. The three main data types and their subtypes are represented by coloured shapes. Simple unlabelled graphs can be represented as binary matrices. Thus, some methods designed for binary data can be applied to them, and vice versa, some graph-mining methods in effect process binary data. The corresponding sections are therefore represented as lying at the intersection between binary and graph data. Dashed and dotted lines are used to group methods associated to the two main strategies (cf. Section 2.3). *Block-based* strategies are used to mine block patterns (also tiles and segments) that group together elements that are similarly distributed. On the other hand, *dictionary-based* strategies are used to mine substructure patterns (also motifs and episodes) that capture specific arrangements and co-occurrences between elements.

2 The basics in a nutshell

The Minimum Description Length (MDL) principle is a model selection method grounded in information theory.

The 1948 article by Shannon is widely seen as the birth certificate of information theory, whereas the introduction of the Minimum Description Length

principle can be dated back to the seminal paper by Rissanen in 1978. The textbooks by Stone (2013) and by Cover and Thomas (2012) provide respectively an accessible introduction and a more detailed account of information theory and related concepts, while the textbook by Grünwald (2007) is often regarded as the major reference about the MDL principle.

The MDL principle is maybe the most popular among several similar principles. It can be seen as a practical variant of the Kolmogorov complexity, according to which the complexity of an object, for instance a dataset, is the length of the shortest computer program that outputs it. The idea is that regularities in the object can be exploited to make the program more concise. For example, the string that consists of a thousand repetitions of the word `baobab` can be produced with the following short program: `for i in range(1000): print('baobab')`. To output instead a string of the same length where the three letters appear in a completely random manner, we have no choice but to embed the string in the program, resulting in a much longer program. The second string is thus considered to be more complex than the first.

However, the Kolmogorov complexity cannot be computed in general. The MDL principle makes it practical by considering more restricted description methods rather than a universal programming language. Specifically, when working with the MDL principle, a class of models of the data is considered and the model that allows to most concisely represent the data is selected.

2.1 Encoding data

Clearly, this is akin to data encoding, where the aim is to map the input object to a sequence, referred to as its *description*, from which the original object can then be reconstructed. In practical scenarios of data encoding, the aim might be to efficiently and reliably either store the object on a storage medium or transmit it over a communication channel. The system of rules that dictates how the object is converted into the description and back is called a *code*. The processes of mapping the object to its description and of reconstructing it are called *encoding* and *decoding*, respectively. The considered storage or channel is typically binary, meaning that the object is mapped to a binary sequence, i.e. over the alphabet $\{0, 1\}$, whose length is hence measured in bits. There has been a lot of studies about communication through noisy channels, that is, when errors might be introduced into the transmitted sequence, and how to recover from it, but this is not of interest to us. Instead of noise-resistant codes, we focus purely on data compression, on obtaining compact descriptions. In general, data compression can be either lossless or lossy, depending whether the source object can be reconstructed exactly or only approximately.

Typically, encoding an object means mapping its elementary parts to binary codewords and concatenating them. Care must be taken to ensure that the resulting bit-string is decodable, that is, that it can be broken down back into the original codewords from which the parts can be recovered. For instance, imagine the data consist of the outcome of five throws of a pair of

dice, i.e. a list of five integers in the interval $[1, 12]$. If we simply turn the values into their binary representations and concatenate them, we might not be able to reconstruct the list of integers. For example, there is no way to tell whether `1110110110` stands for `11 10 1 10 110`, i.e. $\langle 3, 2, 1, 2, 6\rangle$, or for `1 110 1 101 10`, i.e. $\langle 1, 6, 1, 5, 2\rangle$. To avoid this kind of confusion, we want our code to be such that there is a single unambiguous way to split an encoded string into codewords. One strategy is to use separators. For instance, we might represent each integer by as many `1`s, separated by `0`s, so that $\langle 3, 2, 1, 2, 6\rangle$ becomes `1110110101101111110`.

More in general, this is where the *prefix-free property* becomes very useful. A *prefix-free code* (also confusingly known as prefix code or instantaneous code) is such that no extension of a codeword can itself be a codeword.

Fixed-length codes (a.k.a. *uniform codes*), that assign distinct codewords of the same length to every symbol in the input alphabet, clearly satisfy the *prefix-free property*. For an input alphabet consisting of n distinct symbols, the codewords must be of length $\lceil \log_2(n) \rceil$. Such a code minimises the worst-case codeword length, that is, the longest codeword is as short as possible. With such a code, every symbol is worth the same. Therefore it is a good option for pointing out an element among canonically ordered options under a uniform distribution, without a priori bias. For example, the sequence `baeaecdaeeccbc` over the five-letter alphabet $\langle$`a`, `b`, `c`, `d`, `e`$\rangle$ might be encoded in 42 bits, as

`001 000 100 000 100 010 011 000 100 100 010 010 001 010` .

When symbols are not uniformly distributed, using codewords of *different lengths* can result in codes that are more efficient on average. There are only so many short codewords available, so one needs to choose wisely what they are used for. Intuitively, symbols that are more frequent should be assigned shorter codewords. The *Kraft–McMillan inequality* (also known simply as Kraft inequality) gives a necessary and sufficient condition for the existence of a prefix-free code. Specifically, it states that, for a finite input alphabet $\mathcal{A}$, the codeword lengths for any prefix-free code C must satisfy

$$\sum_{x \in \mathcal{A}} 2^{-L_C(x)} \leq 1 \, ,$$

where $L_C(x)$ denotes the length of the codeword assigned by C to symbol x. Vice versa, given codeword lengths satisfying this inequality, there exists a prefix-free code with these codeword lengths. Furthermore, if P is a probability distribution over a discrete input alphabet $\mathcal{A}$, there exists a prefix-free code C such that for all $x \in \mathcal{A}$, $L_C(x) = \lceil -\log_2 \big(P(x)\big) \rceil$. A pair of related techniques to construct such a varying-length prefix-free code is commonly referred to as ***Shannon–Fano code***. Moreover, given an input alphabet $\mathcal{A}$ where each symbol x_i has an associated weight w_i that might represent, in particular, its frequency of occurrence, a code C is optimal if for any other code C' we have

$$\sum_{x_i \in \mathcal{A}} w_i L_C(x_i) \leq \sum_{x_i \in \mathcal{A}} w_i L_{C'}(x_i) \, .$$

Huffman's algorithm is an ingenious simple algorithm allowing to construct an optimal prefix-free code. Considering the example sequence `baeaecdaeeccbc` again, Huffman's algorithm would assign shorter codewords to the more frequent letters `a`, `c` and `e`, while ensuring that the prefix-free property is satisfied, allowing for instance to encode the sequence in just 31 bits, as

```
011 00 11 00 11 10 010 00 11 11 10 10 011 10 .
```

Note that to use such a code, the alphabet and the associated probability distribution must be shared by the sender and the receiver. Indeed, it is not enough that the receiver is able to recover the transmitted codewords, the receiver must also know which symbol is represented by each codeword. In order to transmit a sequence of symbols, a simple way to proceed is therefore to first transmit the information about the distribution, then to transmit the actual sequence using Shannon–Fano coding, resulting in a ***two-part code***. For example, we would first need to transmit the occurrence counts of the five letters, or equivalently their assigned codewords, according to some agreed protocol, before sending the actual encoded 31-bit sequence.

Relying on a sequential prediction, or ***prequential***, strategy provides an alternative for coding that does not require the probability distribution to be known a priori. Simply put, the idea is to start with some initial probability distribution over the alphabet, e.g. uniform, and at each step encode the next element using a prefix-free code based on the current distribution, then update the probability distribution to account for this occurrence. These predictive plug-in codes have useful properties. In particular, the total code length does not depend on the order in which the elements are encoded, and is within a constant factor of the optimal.

Let us consider the sequence `baeaecdaeeccbc` as an example once more. We might start with occurrence counts initialised to one for all five letters, and run Huffman's algorithm to assign them codewords accordingly. Having agreed on this protocol, the sender and the receiver obtain the same initial codewords, without the sender first having to explicitly share the information about the distribution. The sender first transmits the codeword for letter `b`, which the receiver can correctly decode using the initial code. Both sides increment their occurrence count of `b` by one, and update their codewords by running Huffman's algorithm again. Note that at this point, `b` has the highest occurrence count and will thus be assigned a short codeword (2 bits). Next, the sender transmits the new codeword for `a`, which the receiver is able to correctly decode. The occurrence counts are incremented and the codewords updated, on both sides in parallel. And so on, until the entire sequence has been encoded. Crucially, as they apply the same updating protocol in parallel, relying only on information shared so far, sender and receiver maintain identical codes at every step. Later in the process, the occurrence counts of the letters approach their overall frequencies and the lengths of the assigned codewords converge towards those obtained using Shannon–Fano coding with prior knowledge of the distribution.

More recent advances in MDL and model selection theory introduced Bayesian and normalised maximum likelihood (NML) codes. Like prequential codes, both are ***one-part codes***. In contrast to *crude codes*, such *refined codes* remove the need to explicitly encode the parameters of the distribution, thereby avoiding the associated bias, and have useful properties, including optimality guarantees. In particular, the term *universal* is commonly used to refer to a code that, to put it simply, performs essentially as well as the best-fitting code for the input, for any possible input. This use of *universal* should not be confused with another common use, referring to codes for representing integers, which we present next. On the downside, refined one-part codes are not easily explained in terms of practical encoding and are not always computationally feasible.

Finally, a ***universal code for integers*** is a prefix-free code that maps non-negative integers to binary codewords. Such a code can be used to encode positive integers when the upper-bound cannot be determined a priori. *Elias codes*, which come in three variants—namely Elias gamma code, Elias delta code, and Elias omega code—are universal codes commonly used for this purpose. For instance, an integer $x \geq 1$ is encoded using the Elias gamma code as $\lfloor \log_2(x) \rfloor$ zeros followed by the binary representation of x. These codes penalise large values, since larger integers are assigned longer codewords.

2.2 Applying the MDL principle in pattern mining

It is important to note that when applying the MDL principle, **compression is used as a tool to compare models**, rather than as an end in itself. In other words, we do not care about actual descriptions, only about their lengths. Furthermore, we do not care about the absolute magnitude of the description achieved with a particular model as much as compared to those achieved with other candidate models. This has a few important consequences.

First, as the code does not need to be usable in practice, the requirement of integer code lengths can be lifted, allowing for finer comparisons. In particular, this means that for a discrete input alphabet $\mathcal{A}$ with probability distribution P, the most reasonable choice for assigning codewords is a prefix-free code C such that for all $x \in \mathcal{A}$, $L_C(x) = -\log_2\big(P(x)\big)$. This is often referred to as *Shannon–Fano coding*, although no underlying practical code is actually required, or even considered. Note that the entropy of P corresponds to the expected number of bits needed to encode in this way an outcome generated by P.

Second, elements that are necessary in order to make the encoding system work but are common to all candidate models might be left out, since they play no role in the comparison. Third, to ensure a fair comparison, only lossless codes are considered. Indeed, comparing models on the basis of their associated description lengths would be meaningless if we are unable to disentangle the savings resulting from a better ability to exploit the structure of the data, versus from increased distortion in the reconstruction. In some

cases, this simply means that corrections that must be applied to the decoded data in order to achieve perfect reconstruction are considered as part of the description.

Though one-part codes might be used for components, at the highest level most proposed approaches use the *crude* ***two-part MDL***, which requires to first explicitly describe the model M, then describe the data using that model, rather than *refined one-part MDL*, which corresponds to using the entire model class $\mathcal{M}$ to describe the data. That is because the aim is not just to know how much the data can be compressed, but how that compression is achieved, by identifying the associated model. The overall description length is the sum of the lengths of the two parts, so the best model $M \in \mathcal{M}$ to explain the data D is the one that minimises $L(M, D) = L(M) + L(D \mid M)$, where L denotes description lengths in bits, as above. The two parts can be understood as capturing respectively the **complexity of the model** and the **fit of the model to the data**, and the MDL principle as a way to strike a balance between the two.

One can also view this from **the perspective of probabilistic modeling**. Consider a family of probability distributions parameterised by some set Θ, that is, $\mathcal{M} = \{p_\theta, \theta \in \Theta\}$, where each p_θ assigns probabilities to datasets. In addition, consider a distribution π over the elements of $\mathcal{M}$. In this context, in accordance with the Bayesian framework, the best explanation for a dataset D is the element of $\mathcal{M}$ minimising

$$-\log \pi(\theta) - \log p_\theta(D)\,,$$

where the two terms are the prior and the negative log likelihood of the data, respectively. When using a uniform prior, this means selecting the maximum likelihood estimator. Since codes can be associated to probability distributions, we can see the connection to the two-part MDL formulation above, where the term representing the description length of the model, $L(M)$, corresponds to the prior, and the term representing the description length of the data given the model, $L(D \mid M)$, corresponds to the likelihood.

The *Bayesian Information Criterion (BIC)* is a closely related model selection method that aims to find a balance between the fit of the model, measured in terms of the likelihood function, and the complexity of the model, measured in terms of the number of parameters k. Denoting as n the number of data points in D, it can be written as $k \log n - 2 \log p_\theta(D)$.

When applying the MDL principle to a pattern mining task, the models considered consist of patterns, capturing structure and regularities in the data. Depending on the type of data and the application at hand, one must decide what kind of patterns are relevant, i.e. ***(i)*** **a pattern language must be chosen**. The model class $\mathcal{M}$ then consists of all possible subsets of patterns from the chosen pattern language. Note that we generally use the term *model* to refer to a specific collection of patterns and the single corresponding probability distribution over the data, whereas from the statistical modeling perspective, *model* typically refers to a family of probability distributions. Next, ***(ii)*** **an**

encoding scheme must be designed, i.e. a mechanism must be engineered to encode patterns of the chosen type and to encode the data by means of such patterns. Finally, ***(iii)*** **a search algorithm must be devised** to explore the space of patterns and identify the best set of patterns with respect to the encoding, i.e. the set of patterns that results in the shortest description length.

2.3 Dictionary-based vs. block-based strategies

The crude two-part MDL requires to explicitly specify the model (the $L(M)$ part). This means designing an ad-hoc encoding scheme to describe the patterns. This is both a blessing and a curse, because it gives leeway to introduce some bias, i.e. penalise properties of patterns that are deemed less interesting or useful than others by making them more costly in terms of code length. But it can involve some difficult design choices and lead to accusations of "putting your fingers on the scale".

When considering substructure patterns, encoding the data using the model (the $L(D \mid M)$ part) can be fairly straightforward, simply replacing occurrences of the patterns in the data by their associated codewords. Knowing how many times each pattern X is used in the description of the data D with model M, denoted as $\text{usage}_{D,M}(X)$, X can be assigned a codeword of length

$$L(X) = -\log_2 \left(\frac{\text{usage}_{D,M}(X)}{\sum_{Y \in M} \text{usage}_{D,M}(Y)} \right)$$

using Shannon–Fano coding. The design choice of how to cover the data with a given set of patterns, dealing with possible overlaps between patterns in M, determines where each pattern is used, defining $\text{usage}_{D,M}(X)$. The requirement that the encoding be lossless means that elementary patterns, e.g. singleton itemsets, must be included in the model, to ensure complete coverage. In this case, encoding the model (the $L(M)$ part) consists in providing the mapping between patterns and codewords, typically referred to as the code table. That is, for each pattern in turn, describing it and indicating its associated codeword. Such a *code table* or ***dictionary-based*** **strategy**, which corresponds to frequent pattern mining approaches, is a common way to proceed.

An alternative strategy is to divide the data into blocks, that might or might not be allowed to overlap, each of which is associated with a specific probability distribution over the possible values and should be as homogeneous as possible. The values within each block are then encoded using a code optimised for the corresponding distribution. In that case, encoding the model consists in indicating the limits of each block and the associated probability distribution. Such a ***block-based*** **strategy** corresponds to segmentation, biclustering and tiling approaches.

These two main strategies, *dictionary-based* and *block-based*, that use respectively substructure and block patterns, are an important distinguishing factor that we use to categorise approaches, as depicted in Fig. 1 with dotted

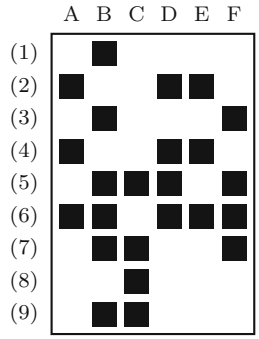


Fig. 2 A toy binary dataset, with six columns (i.e. items) denoted A–F, nine rows (i.e. transactions) denoted (1)–(9), containing twenty-four ones (i.e. positive entries or item occurrences) represented as black squares.

and dashed lines, respectively. We further illustrate and contrast these strategies on a toy binary dataset shown in Fig. 2. The dataset has six columns (i.e. items) denoted A–F, nine rows (i.e. transactions) denoted (1)–(9), and contains twenty-four ones (i.e. positive entries or item occurrences) represented as black squares. In particular, the approaches delineated here to illustrate the two strategies are based on the work of Siebes et al. (2006) and Smets and Vreeken (2012) (cf. Section 3), on one hand, and of Chakrabarti et al. (2004) (cf. Section 4.2), on the other hand.

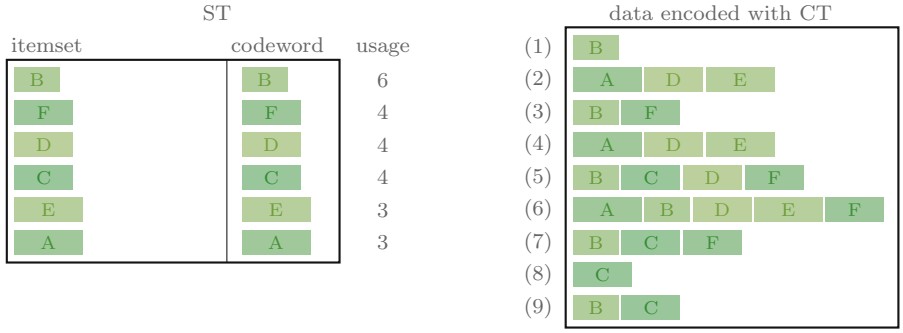
Examples of **modeling the toy binary dataset using a dictionary-based approach** are provided in Fig. 3. The simplest model is considered first *(i)*, which contains as its patterns all singleton itemsets from the dataset and is often referred to as the *standard code table (ST)*. A non-trivial code table is considered next *(ii)*. In both cases, the model is represented as a code table associating patterns, here itemsets, to codewords. Encoding the model means encoding each pair. On one hand, itemsets are specified by listing the items they contain. This uses Shannon–Fano coding over the alphabet of items associated to their frequency of occurrence in the data, which is assumed to be shared knowledge. On the other hand, codewords are assigned using Shannon–Fano coding over the alphabet of itemsets included in the code table, associated to their usage.

In Fig. 3, the prefix-free codewords assigned to items and to itemsets are represented by coloured blocks in shades of green and blue, respectively. The width of a block is proportional to the length of the represented codeword. For instance, the first row of the non-trivial code table (bottom-left quadrant of Fig. 3) specifies the first itemset in the code table, in this case ADE, listing the three items that constitute it, A, D and E, using the items' codewords (green blocks), then provides the codeword assigned to ADE by the code based on the usage of itemsets selected in this model (blue block).

Note that for the standard code table, these two prefix-free codes are the same because the standard code table consists of all singleton items and their

$L(M, D) = \quad L(M) \quad + \quad L(D \mid M)$

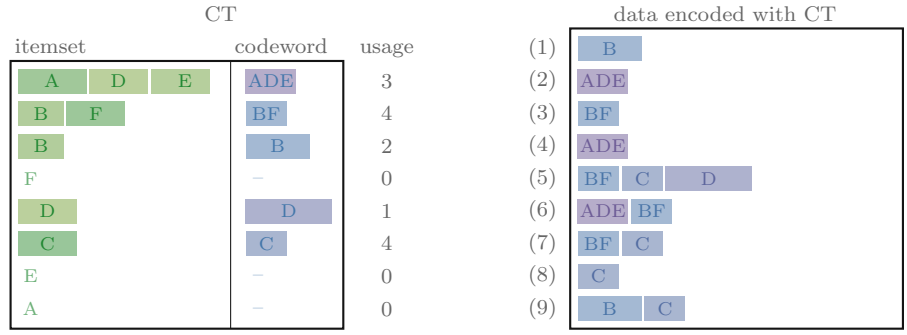
i) The simplest model, with all singleton itemsets, a.k.a. *standard code table (ST).*



$$
\begin{aligned}
&-\log_2(6/24) - \log_2(6/24)\\
&-\log_2(4/24) - \log_2(4/24)\\
&-\log_2(4/24) - \log_2(4/24)\\
&-\log_2(4/24) - \log_2(4/24)\\
&-\log_2(3/24) - \log_2(3/24)\\
&\underbrace{-\log_2(3/24)}_{\text{itemset}} \underbrace{- \log_2(3/24)}_{\text{codeword}}
\end{aligned}
\qquad
\begin{aligned}
&-6 \cdot \log_2(6/24)\\
&-4 \cdot \log_2(4/24)\\
&-4 \cdot \log_2(4/24)\\
&-1 \cdot \log_2(4/24)\\
&-3 \cdot \log_2(3/24)\\
&\underbrace{-3 \cdot \log_2(3/24)}_{\text{listing itemsets occurrences}}
\end{aligned}
$$

$$= \quad 31.510 \qquad + \quad 61.020 \quad = \quad 92.530$$

ii) A non-trivial code table (CT).



$$
\begin{aligned}
&-\log_2(4/24) - 2 \cdot \log_2(3/24) - \log_2(3/14)\\
&-\log_2(6/24) - \log_2(4/24) - \log_2(4/14)\\
&-\log_2(6/24) - \log_2(2/14)\\
&-\log_2(4/24) - \log_2(1/14)\\
&\underbrace{-\log_2(4/24)}_{\text{itemset}} \underbrace{- \log_2(4/14)}_{\text{codeword}}
\end{aligned}
\qquad
\begin{aligned}
&-3 \cdot \log_2(3/14)\\
&-4 \cdot \log_2(4/14)\\
&-2 \cdot \log_2(2/14)\\
&-1 \cdot \log_2(1/14)\\
&\underbrace{-4 \cdot \log_2(4/14)}_{\text{listing itemsets occurrences}}
\end{aligned}
$$

$$= \quad 32.790 \qquad + \quad 30.543 \quad = \quad 63.333$$

Fig. 3 Dictionary-based strategy, examples on the toy binary dataset of Fig. 2. The model M consists of a code table associating patterns, here itemsets, to codewords (left). The data D is encoded using the model by replacing occurrences of the itemsets by the associated codewords (right). Coloured blocks represent prefix-free codewords assigned to items (green) and itemsets (blue), their width is proportional to the code length.

usage is precisely their frequency of occurrence in the data. Therefore, all codewords in the top half of Fig. 3 are represented as green blocks. For the same reason, the length of the codeword associated to item x by the first code, which does not depend on the code table, is denoted as $L_{ST}(x)$, whereas the length of the codeword associated to itemset X by the second code, which is different for different code tables, is denoted as $L_{CT}(X)$.

Then, encoding the data using the model simply means replacing itemset occurrences by the corresponding codewords.

In summary, the overall description length can be computed as

$$\begin{aligned} L(M, D) &= L(M) + L(D \mid M) \\ &= \sum_{X \in M} \underbrace{\sum_{x \in X} L_{ST}(x)}_{\text{itemset}} + \underbrace{L_{CT}(X)}_{\text{codeword}} + \underbrace{\sum_{X \in M} \text{usage}(X) \cdot L_{CT}(X)}_{\text{listing itemsets occurrences}} . \end{aligned} \quad (1)$$

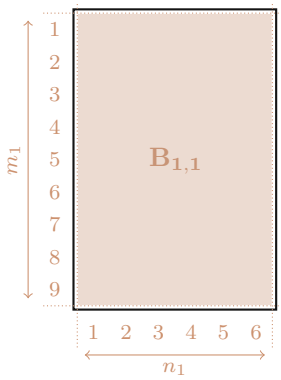
In this example, the standard code table and the non-trivial code table yield an overall description length of 92.530 bits and 63.333 bits, respectively. By identifying items that occur frequently together, the latter results in a shorter description length, and one can therefore conclude that it constitutes a better model for the dataset, according to the MDL criterion.

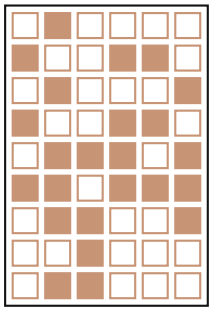
A similar approach can be built by replacing static Shannon–Fano coding with dynamic prequential coding. In that case, encoding the model only requires to list the itemsets, i.e. only the first column of the code table is needed. Then, the data is encoded by enumerating the itemset occurrences using prequential coding (cf. Section 2.1). In practise, the process of updating the codewords does not actually need to be carried through. Luckily, the description length of the data can be calculated with a (not so simple) formula that does not depend on the order in which the itemsets are transmitted (see Budhathoki and Vreeken, 2015).

Examples of **modeling the toy binary dataset using a block-based approach** are provided in Fig. 4. The simplest model is considered first *(i)*, which consists of a single block. A non-trivial model dividing the dataset into six blocks is considered next *(ii)*. The approach illustrated here requires the patterns to form a partition of the data. Therefore, any considered model consists of a set of non-overlapping blocks covering the entire dataset and, more specifically, is such that the rows and the columns are divided into disjoint groups. This requirement means that each row and each column belongs to exactly one group, a fact that can be exploited when designing the encoding, as explained below. Each block is associated to a specific probability distribution over the values occurring within it. In this case, the values are zero and one, corresponding to a Bernoulli distribution. Encoding the model means specifying the groups of rows and of columns that define the blocks, and the probability associated to each one. In this approach, all blocks are induced by a unique partition of the rows and of the columns, which can be specified by defining an order over the rows (resp. columns) and indicating the number

$L(M, D) =$ $L(M)$ $+$ $L(D \mid M)$

i) The simplest model, with a single block.





$\underbrace{\mathrm{L}_{\mathbb{N}}(9)}_{m=9 \text{ rows}} + \underbrace{9 \cdot \log_2(9)}_{<1,2,3,4,5,6,7,8,9>} + \underbrace{\mathrm{L}_{\mathbb{N}}(6)}_{n=6 \text{ cols}} + \underbrace{6 \cdot \log_2(6)}_{<1,2,3,4,5,6>}$

$+ \underbrace{\mathrm{L}_{\mathbb{N}}(1)}_{k=1 \text{ row group}} + \underbrace{0}_{(<9>)} + \underbrace{\mathrm{L}_{\mathbb{N}}(1)}_{l=1 \text{ col group}} + \underbrace{0}_{(<6>)}$

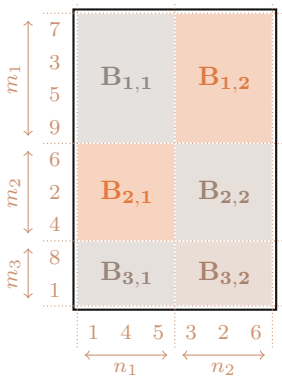
$+ \underbrace{\log_2(9 \cdot 6 + 1)}_{24 \text{ ones in } B_{1,1}}$

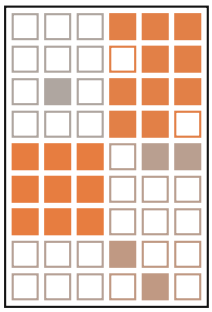
$= 64.686$

$\underbrace{-30 \cdot \log_2(30/54) - 24 \cdot \log_2(24/54)}_{B_{1,1}:<0,1,0,\ldots,0,1,1,0,0,0>}$

$+ 53.518 = 118.204$

ii) A non-trivial model partitioning the dataset into six blocks.





$\underbrace{-11 \cdot \log_2(11/12) - 1 \cdot \log_2(1/12)}_{B_{1,1}:<0,0,0,0,0\ldots0,0,0,0>}$

$\underbrace{-2 \cdot \log_2(2/12) - 10 \cdot \log_2(10/12)}_{B_{1,2}:<1,1,1,0\ldots1,1,1,0>}$

$\underbrace{\mathrm{L}_{\mathbb{N}}(9)}_{m=9 \text{ rows}} + \underbrace{9 \cdot \log_2(9)}_{<7,3,5,9,6,2,4,8,1>} + \underbrace{\mathrm{L}_{\mathbb{N}}(6)}_{n=6 \text{ cols}} + \underbrace{6 \cdot \log_2(6)}_{<1,4,5,3,2,6>}$

$\underbrace{-0 \cdot \log_2(0/9) - 9 \cdot \log_2(9/9)}_{B_{2,1}:<1,1,1,1,1,1,1,1,1>}$

$+ \underbrace{\mathrm{L}_{\mathbb{N}}(3)}_{k=3 \text{ row groups}} + \underbrace{\log_2(7) + \log_2(4)}_{<4,3(,2)>} + \underbrace{\mathrm{L}_{\mathbb{N}}(2)}_{l=2 \text{ col groups}} + \underbrace{\log_2(5)}_{<3(,3)>}$

$\underbrace{-7 \cdot \log_2(7/9) - 2 \cdot \log_2(2/9)}_{B_{2,2}:<0,1,1,0,0,0,0,0,0>}$

$+ \underbrace{\log_2(4 \cdot 3 + 1)}_{1 \text{ one in } B_{1,1}} + \underbrace{\log_2(4 \cdot 3 + 1)}_{10 \text{ ones in } B_{1,2}} + \underbrace{\log_2(3 \cdot 3 + 1)}_{9 \text{ ones in } B_{2,1}}$

$\underbrace{-6 \cdot \log_2(6/6) - 0 \cdot \log_2(0/6)}_{B_{3,1}:<0,0,0,0,0,0>}$

$+ \underbrace{\log_2(3 \cdot 3 + 1)}_{2 \text{ ones in } B_{2,2}} + \underbrace{\log_2(2 \cdot 3 + 1)}_{0 \text{ one in } B_{3,1}} + \underbrace{\log_2(2 \cdot 3 + 1)}_{2 \text{ ones in } B_{3,2}}$

$\underbrace{-4 \cdot \log_2(4/6) - 2 \cdot \log_2(2/6)}_{B_{3,2}:<1,0,0,0,1,0>}$

$= 88.279$

$+ 25.154 = 113.433$

Fig. 4 Block-based strategy, examples on the toy binary dataset of Fig. 2. The model M partitions the dataset D into blocks, each associated to a specific probability distribution over the values (left), which is used to encode the entries (right). More intense shades of orange represent higher probabilities of ones within the corresponding block.

of rows (resp. columns) in each group. Finally, the number of ones in each block is transmitted, allowing to compute the corresponding probability. This is achieved through a combination of fixed-length and universal codes.

Next, we delve deeper into the details of the encoding scheme, to illustrate the choices that are involved in its design. We let

- m and n denote respectively the number of rows and columns in the dataset,
- k and l denote respectively the number of row and column groups,
- m_i and n_j denote respectively the number of rows and columns in block $B_{i,j}$,
- $\gamma_v(B_{i,j})$ denote the number of entries in block $B_{i,j}$ equal to v, and
- $\mathrm{L}_{\mathbb{N}}(x)$ denote the MDL optimal universal code length for integer x.[3]

The formula for computing the overall description length is provided in Equation 2. The number of rows and the number of columns are transmitted using universal coding, since these values are not bounded a priori, and the order of rows (resp. of columns) is then specified by listing row (resp. column) identifiers in turn, using a fixed-length code (cf. part (a) of Equation 2). In fact, this part of the encoding is independent of the model and has a constant length for a given dataset. Therefore, it does not impact the comparison and can be ignored. The number of row groups k (resp. column groups l) could be transmitted using a fixed-length code $\log_2(m)$ (resp. $\log_2(n)$), since there cannot be more groups than there are rows (resp. columns). However, universal coding is used instead, to favour partitions with fewer groups. Then, assuming that the numbers of rows and of columns in the groups are sorted and transmitted by decreasing order, upper bounds m_i^* and n_j^* on m_i and n_j, respectively, can be derived given shared knowledge since already transmitted values constrain the remaining ones: $m_i^* = \left(\sum_{t=i}^{k} m_t\right) - k + i$, for $i = 1, \dots k-1$ and $n_j^* = \left(\sum_{t=j}^{l} n_t\right) - l + j$, for $j = 1, \dots l-1$.

These upper bounds are used to transmit the numbers of rows and of columns in the groups with fixed-length codes (cf. part (b) of Equation 2). Also the number of ones in each block can be transmitted using a fixed-length code, since it takes value between zero and the number of entries in the block (cf. part (c) of Equation 2).

The data is then encoded using the model. This is done by listing the entries in each block with a prefix-free code adjusted to the probability distribution within the block (cf. part (d) of Equation 2).

[3] $\mathrm{L}_{\mathbb{N}}(x)$ is defined for $x > 1$ as $\mathrm{L}_{\mathbb{N}}(x) = \log_2(x) + \log_2\log_2(x) + \cdots + \log_2(c_0)$, with the value of c_0 set so as to satisfy the Kraft inequality. Specifically, we let $c_0 = 2.86507$.

In summary, the overall description length can be computed as

$$
\begin{aligned}
L(M, D) = \quad & L(M) + L(D \mid M) \\
= \quad & \underbrace{\underbrace{\mathrm{L}_{\mathbb{N}}(m)}_{\text{nb. rows}} + \underbrace{m \cdot \log_2(m)}_{\text{ordered rows}} + \underbrace{\mathrm{L}_{\mathbb{N}}(n)}_{\text{nb. cols}} + \underbrace{n \cdot \log_2(n)}_{\text{ordered cols}}}_{(a)} \\
& + \underbrace{\underbrace{\mathrm{L}_{\mathbb{N}}(k)}_{\text{nb. row groups}} + \underbrace{\sum_{i=1}^{k-1} \log_2(m_i^*)}_{\text{nb. rows in each group}}}_{} \\
& + \underbrace{\underbrace{\mathrm{L}_{\mathbb{N}}(l)}_{\text{nb. col groups}} + \underbrace{\sum_{j=1}^{l-1} \log_2(n_j^*)}_{\text{nb. cols in each group}}}_{(b)} \\
& + \underbrace{\underbrace{\sum_{i=1}^{k} \sum_{j=1}^{l} \log_2(m_i \cdot n_j + 1)}_{\text{nb. ones in each block}} + }_{(c)} \\
(d) \quad & \underbrace{\sum_{i=1}^{k} \sum_{j=1}^{l} \sum_{v \in \{0,1\}} -\gamma_v(B_{i,j}) \cdot \log_2 \left(\frac{\gamma_v(B_{i,j})}{m_i \cdot n_j} \right)}_{\text{listing entries in each block}} .
\end{aligned}
\tag{2}
$$

In Fig. 4, shades of orange are used to represent probability distributions within the blocks, with more intense shades representing higher probabilities of ones. In this example, the single-block model and the six-block model yield an overall description length of 118.204 bits and 113.433 bits, respectively. By partitioning the dataset into blocks that are particularly dense or particularly sparse, the latter results in a shorter description length, and one can therefore conclude that it constitutes a better model for the dataset, according to the MDL criterion.

Removing the partition constraint and allowing overlaps between the block patterns would require to modify the encoding since we could no longer assume that each row and each column belongs to a single group.

2.4 Algorithms

The MDL principle provides a basis for designing a score for patterns, but no way to actually find the best collection of patterns with respect to that score. The space of candidate patterns, and even more so the space of candidate pattern sets, is typically extremely large, if not infinite, and rarely possesses any useful structure. Hence, exhaustive search is generally infeasible, heuristic search algorithms are employed, and one must be satisfied with finding a good set of patterns rather than an optimal one. Mainly, algorithms can be divided between *(i)* approaches that generate a large collection of patterns then (iteratively) select a small set out of it, and *(ii)* approaches that generate candidates

on-the-fly, typically in a levelwise manner, possibly in an anytime fashion. The former approaches are typically less efficient, since they generate many more candidates than necessary, but constitute a useful basis for building a proof-of-concept. Because recomputing costs following the addition or removal of a candidate pattern is often prohibitively expensive, an important component of the heuristic search algorithms consists in efficiently and accurately bounding these costs.

With a dictionary-based strategy, the simplest model consists of all separate basic elements. The search for candidates can thus start from this model and progressively combine elements. Vice versa, with a block-based strategy, the simplest model consists of a single block covering the entire dataset. The search for candidates can thus start from this model and progressively split elements. Intuitively, the two main strategies lend themselves respectively to bottom-up and top-down iterative exploration algorithms.

In summary, to apply the MDL principle to a pattern mining task, one might proceed in three main steps. First, define the pattern language, deciding what constitutes a structure of interest given the data and application considered. Second, define a suitable encoding scheme, designing a system to encode the patterns and to encode the data using such patterns. Third, design a search algorithm, allowing to identify in the data a collection of patterns that yields a good compression under the chosen encoding scheme.

Our main focus here is on the second step, the design of an encoding scheme for the different types of patterns, which is the core of the MDL methodology and its distinctive ingredient. For the most part, we do not discuss search algorithms and are not concerned by issues of performance, which are more generic aspects of pattern mining methodologies.

3 Mining itemsets with Krimp & Co.

A transactional database consists of a collection of sets, called transactions, over a universe of items. The prototypical example for this type of data comes from market-basket analysis, which is also where some of the terminology is borrowed from. Alternatively, a transactional database can be represented as a binary matrix. Frequent itemset mining, that is, finding items that frequently co-occur in a transactional database, is a central task in data mining (cf. Section 1).

3.1 Krimp

The introduction by Siebes et al. in 2006 of a MDL-based algorithm for mining and selecting small but high-quality collections of itemsets sparked a productive line of research, including algorithmic improvements, adaptations to different tasks, and various applications of the original algorithm. The algorithm,

soon dubbed Krimp (van Leeuwen et al., 2006), Dutch for "to shrink", is a prime example of a *dictionary-based* strategy (cf. Section 2.3), illustrated in Fig. 3.

Through an evaluation on a classification task, van Leeuwen et al. (2006) show that the selected itemsets are representative. Specifically, considering a labelled training dataset, Krimp is applied separately on the transactions associated to each class to mine a code table. A given test transaction can then be encoded using each of the code tables, and assigned to the class that corresponds to the shortest code length.

The Krimp algorithm has subsequently been presented with varying depths of technicality (van Leeuwen et al., 2009b; Vreeken et al., 2011; Siebes, 2014; van Leeuwen and Vreeken, 2014). It is a corner stone of a couple of doctoral dissertations (Vreeken, 2009; van Leeuwen, 2010) and also contributed to spark a more conceptual discussion (Siebes, 2012).

3.2 Algorithmic improvements

Works building on Krimp include several algorithmic improvements. In particular, the Slim algorithm of Smets and Vreeken (2012) modifies Krimp and greedily generates candidates by merging patterns, instead of evaluating candidates from a pre-mined list. The example presented in Fig. 3*(ii)* (cf. Section 2.3) was actually obtained by running the Slim algorithm on the toy dataset of Fig. 2. Hess et al. (2014) propose a data structure similar to the FP-tree to facilitate the recomputation of usages when the Krimp code table is updated, making the mining algorithm more efficient. Sampson and Berthold (2014) apply widening, i.e. diversification of the search, to Krimp.

3.3 Finding differences and anomalies

The analysis of differences between databases and the detection of anomalies are derivative tasks that have attracted particular attention. Vreeken et al. (2007a) use Krimp to measure differences between two databases, by comparing the length of the description of a database obtained with a code table induced on that database versus one induced on the other database. The coverage of individual transactions by the selected itemsets, and the specific code tables obtained are also compared. The DiffNorm algorithm introduced by Budhathoki and Vreeken (2015) aims to encode multiple databases at once without redundancy, and allows to investigate the differences and similarities between the databases by inspecting the obtained code table. As a major contribution, Budhathoki and Vreeken (2015) improve the encoding by replacing the Shannon–Fano code used in the original Krimp by a prequential plug-in code (cf. Section 2.1).

Smets and Vreeken (2011) use Krimp for outlier detection by looking at how many bits are needed to encode a transaction. If this number is much

larger than expected, the transaction is declared anomalous. In addition, the encodings of transactions can be scrutinised to obtain further insight into how they depart from the rest. Akoglu et al. (2012b) design an algorithm that detects as anomalies transactions having a high encoding cost. Their proposed algorithm mines multiple code tables, rather than a single one in KRIMP, and handles categorical data. Bertens et al. (2015) (also Bertens et al., 2017) propose a method to detect anomalous co-occurrences based on the KRIMP/SLIM code tables.

3.4 Mining rule sets

Going beyond itemsets, a closely related task is to mine rules. van Leeuwen and Galbrun (2015) propose to mine association rules across a two-view transactional dataset, such that one view can be reconstructed from the other, and vice versa. Fischer and Vreeken (2019) instead consider a unique set of items and aim to mine associations rules that allow to reconstruct the dataset, enabling corrections in order to increase the robustness of the results. They then apply the approach to learn rules about activation patterns in neural networks (Fischer et al., 2021).

Aoga et al. (2018) present a method to encode a binary label associated to each transaction, using the original transactions and a list of rules, each associated to a probability that the target variable holds true. Proença and van Leeuwen (2020a) (also Proença and van Leeuwen, 2020b) consider a similar task, but with multiple classes and targeted towards predictive rather than descriptive rules, then looking for rules that capture deviating groups of transactions, i.e. dealing with the subgroup discovery task (Proença et al., 2020, 2021b). Beside binary attributes, i.e. items, Proença et al. (2020) also consider nominal and numerical attributes, and aim to predict a single numerical target, modeled using normal distributions, instead of a binary target. Proença et al. (2021a) further extend the approach to more complex nominal and numerical targets by resorting to normalised maximum likelihood (NML) and Bayesian codes, respectively.

Whereas the former two output a set of rules, these latter methods return a list of rules, such that at most one rule, the first valid rule encountered in the list, applies to any given transaction. In all cases, the dataset, or part of it, is assumed to be given and the aim is to reconstruct a target variable, or the rest of the dataset.

3.5 Other adaptations

Further work also includes extending the KRIMP approach to more expressive patterns, such as patterns mined from relational databases (Koopman and Siebes, 2008, 2009), and using KRIMP for derivative tasks.

van Leeuwen and Siebes (2008) use KRIMP to detect changes in the distribution of items in a streaming setting. If a code table induced from an earlier

part of the stream no longer provides good compression as compared to a code table induced from a more recent part of the stream, it is a signal that the distribution has changed. Bonchi et al. (2011) extend the approach to the probabilistic setting, where the occurrence of an item in a transaction is associated to a probability, aiming to find a collection of itemsets that compress the data well in expectation.

Given an original database, Vreeken et al. (2007b) use Krimp to generate a synthetic database similar to the original one, for use in the context of privacy-preserving data mining. The code table is induced on the original database, itemsets are then sampled from it and combined to generate synthetic transactions. Vreeken and Siebes (2008) use Krimp for data completion. In a way, this turns the MDL principle on its head. Starting from an incomplete database, rather than looking for the patterns that compress the data best, the proposed approach looks for the data that is best compressed by the patterns and considers that to be the best completion. Instead of a single code table, Siebes and Kersten (2011) look for a collection of code tables, that capture the structure of the dataset at different levels of granularity. Representing a categorical dataset as transactional data by mapping each value of an attribute to a distinct item implies that each transaction contains exactly one item for each categorical attribute, and hence that all transactions have the same length. Siebes and Kersten (2012) consider the problem of smoothing out the small scale structure from such datasets. That is, they aim to replace entries in the data so that its large scale structure is maintained but it can be compressed better.

3.6 Applications

The Krimp algorithm has also been employed to tackle problems in different domains, including clustering tagged media (van Leeuwen et al., 2009a), summarising text (Vanetik and Litvak, 2017, 2018), detecting malware (Asadi and Varadharajan, 2019a,b) and analysing the Semantic Web (Bobed et al., 2019).

4 Tabular data (continued)

Transactional data can be seen as a binary matrix or table, and is hence a form of tabular data. Data tables might also contain categorical or real-valued attributes, which can be either turned into binary attributes as a preprocessing or handled directly with dedicated methods.

4.1 More itemsets

Beside Krimp and algorithms inspired from it, different approaches have been proposed to mine itemsets from binary matrices using the MDL principle.

Heikinheimo et al. (2009) focus on finding collections of low-entropy itemsets, typically even more compact than those obtained with Krimp. On the other hand, the main objective of Mampaey and Vreeken (2010) is to provide a summary of the dataset in the form of a partitioning of the items. For each partition, codewords are associated to the different combinations of the items that comprise the partition, and used to encode the corresponding subset of the data. In the Pack algorithm, proposed by Tatti and Vreeken (2008), the code representing an item is made dependent on the presence/absence of other items in the same transaction. This can be represented as decision trees whose intermediate nodes are items and leaves contain code tables for other items. Fischer and Vreeken (2020) introduce a rich pattern language that can capture both co-occurrences and mutual exclusivity of items.

Mampaey et al. (2012) present a variant of their MTV itemset mining algorithm (Mampaey et al., 2011) where the MDL principle is used to choose the collection of itemsets that yields the best model, as an alternative to the Bayesian Information Criterion (BIC).

4.2 Blocks in binary data

A different type of patterns that can be mined from binary matrices consists of blocks defined by a collection of rows and columns with a homogeneous density of ones, sometimes known as *tiles* or as *biclusters*, constituting the basis of *block-based* strategies (cf. Section 2.3). Chakrabarti et al. (2004) propose a method to partition the rows and columns of the matrix into subsets that define homogeneous blocks. The example presented in Fig. 4 follows this approach. Tatti and Vreeken (2012a) introduce the Stijl algorithm to mine a hierarchy of tiles from a binary matrix. The adjectives *geometric* or *spatial* typically refer to cases where the order of the rows and columns of the data matrix is meaningful (e.g. Papadimitriou et al., 2005; Faas and van Leeuwen, 2020).

4.3 Formal Concept Analysis (FCA)

The field of Formal Concept Analysis focuses on basically the same problem as itemset mining, but from a slightly different perspective and with its own formalism. Some steps have been taken towards applying and further developing MDL-based methods within this framework (Otaki and Yamamoto, 2015; Yurov and Ignatov, 2017). In particular, as a topic in her doctoral dissertation, Makhalova (2021) extensively studied MDL-based itemset mining methods from the perspective of FCA (also Makhalova et al., 2018a,b, 2019b, 2021).

4.4 Boolean Matrix Factorisation (BMF)

Factorising a matrix consists in identifying two matrices, the factors, so that the original matrix can be reconstructed as their product. One challenge is to find the right balance between the complexity of the decomposition (generally measured by the size of the factors) and the accuracy of the reconstruction, which is where the MDL principle comes in handy (Miettinen and Vreeken, 2011, 2014; Hess et al., 2017). Whereas the other approaches permit reconstruction errors in both values, Makhalova and Trnecka (2019) do not allow factors to cover zero entries, considering so-called "from-below" factorisations (also Makhalova and Trnecka, 2021). Lucchese et al. (2014) (also Lucchese et al., 2010b,a) propose a MDL-based score to compare factors of a fixed size.

An example of a factorisation of the toy binary dataset from Fig. 2 is provided in Fig. 5. The model consists of a pair of factor matrices whereas the data is encoded as a mask of corrections. To reconstruct the original data, the Boolean matrix product of the factors is computed and the mask of corrections is applied to the resulting matrix.

When applied to a Boolean matrix, factorisation shares some similarities with itemset mining, as it aims to identify items that occur (or do not occur) frequently together. That is, the two factor matrices can be interpreted as specifying itemsets and indicators of occurrence, respectively. On the other hand, the factors can also be interpreted as specifying possibly overlapping fully dense blocks. A mask applied to the reconstructed data provides a global error correction mechanism. Thus, Boolean matrix factorisation can be seen as a hybrid approach.

4.5 Categorical data

One way to mine datasets involving categorical attributes is to binarise them and then apply, for instance, an itemset mining algorithm. However, binarisation entails a loss of information and dedicated methods can hence offer a better alternative.

Mampaey and Vreeken (2013) introduce an approach to detect correlated attributes, i.e. such that the different categories occur in a coordinated manner, whereas He et al. (2014) present a subspace clustering method, i.e. look not only for groups of attributes, but also corresponding groups of rows where coordinated behaviour occurs.

4.6 Numerical data

A numerical data table containing m columns can be seen as a collection of points in a m-dimensional space. In pattern mining, numerical data is often handled by applying discretisation, which requires to partition the data into coherent blocks. While it can be seen as a data exploration task in its own

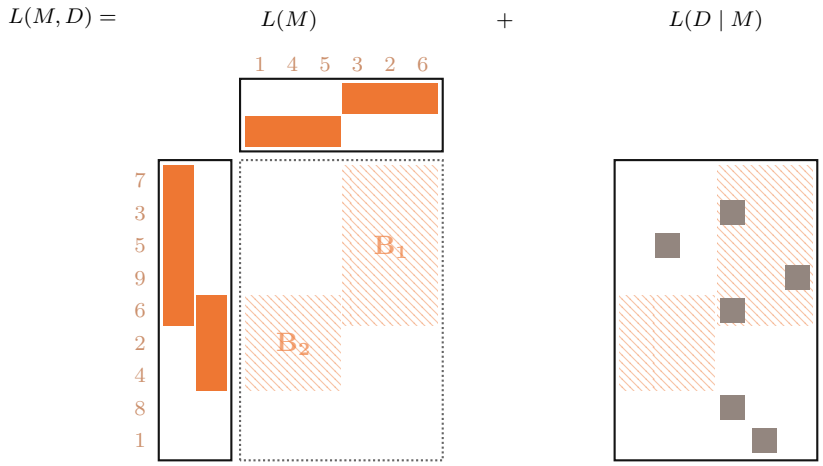


Fig. 5 Boolean Matrix Factorisation, example on the toy binary dataset of Fig. 2. The model M consists of a pair of factor matrices (left) and the data D is encoded as a mask of corrections (right). To reconstruct the original data, first the Boolean matrix product of the factors is computed, resulting in the matrix with two fully dense blocks B_1 and B_2. Then, the mask of corrections is applied to this matrix, that is, the value of the cells indicated in the mask (grey squares) are flipped. Note that the blocks drawn with the hatch pattern depict the intermediate step of the reconstruction and are not actually encoded as such.

right, providing an overview of the dataset and the distribution of values, discretisation often constitutes a pre-processing task to allow applying algorithms that can handle only discrete input data. Yet, choosing good parameters for the discretisation can be difficult, and its quality can have a major impact on later processing. Unsupervised discretisation, where no side information is available, is in contrast to supervised discretisation, that takes into account class labels and often precedes a machine learning task. Here we focus on the former.

Kontkanen and Myllymäki (2007) propose a histogram density estimation method that relies on the MDL principle, formalised using the normalised maximum likelihood (NML) distribution. This method is employed by Kameya (2011) to discretise time-series seen as a collection of two-dimensional time–measurement data points, and extended by Yang et al. (2020) to two-dimensional numerical data, more in general. Along similar lines, Nguyen et al. (2014) aim to automatically identify a high-quality discretisation that preserves the interactions between attributes. Witteveen et al. (2014) extend the Kraft inequality (cf. Section 2.1) to numerical data and introduce an approach to find hyperintervals, i.e. multidimensional blocks.

Makhalova et al. (2019a) consider the problem of mining interesting hyperrectangles from discretised numerical data, and aim to design an encoding that accommodates overlaps between patterns (Makhalova et al., 2020, 2022).

Lakshmanan et al. (2002) formalise mining OLAP data, i.e. multidimensional datasets, as a problem of finding a cover in a multidimensional array

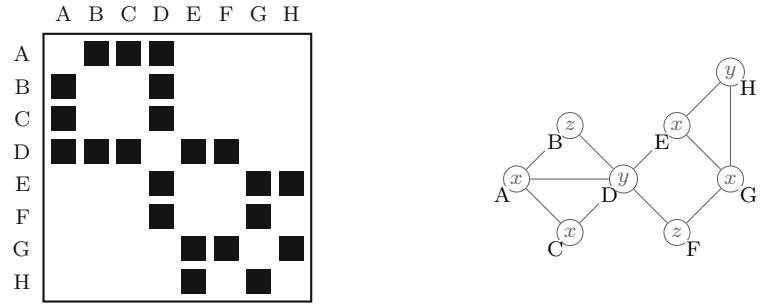


Fig. 6 A toy graph, with eight nodes denoted *A*–*H* carrying labels from $\{x, y, z\}$ and connected by undirected edges (right). The adjacency matrix (left) captures the structure of the graph, ignoring the labels.

containing positive, negative and neutral cells. The aim is then to find the most compact set of hyperrectangles that covers all positive cells, none of the negative cells, and no more than a chosen number of the neutral cells. The score is presented as a generalised MDL due to the tolerance on neutral cells. However, coverings are evaluated by simply counting cells, which does not actually adhere with the principle, generalised or otherwise.

5 Graphs

In this section, we consider approaches for mining graphs. At their simplest, graphs are undirected and unlabelled, but they can also come with directed edges, with node or edge labels, or be dynamic, that is, time-evolving. The main tasks consist in identifying nodes that have similar connection patterns to group them into homogeneous blocks and in finding recurrent connection substructures. These correspond respectively to *block-based* and *dictionary-based* strategies (cf. Section 2.3).

For illustrative purposes, we consider a toy graph, shown in Fig. 6, and delineate approaches that follow either strategy. The example shown in Fig. 7 illustrates the *block-based* strategy and follows the work of Chakrabarti (2004) (cf. Section 5.1), whereas the example shown in Fig. 8 illustrates the *dictionary-based* strategy and follows the work of Bariatti et al. (2020a) (cf. Section 5.5).

Looking at the corresponding adjacency matrix, a simple unlabelled graph can be represented as a binary table. Approaches from Sections 3 and 4 can thus readily be used for mining graphs. On one hand, the problem of grouping nodes into blocks that constitute particularly dense subgraphs, or communities, is closely related to identifying particularly dense tiles in a binary matrix. On the other hand, approaches that follow a *dictionary-based* strategy and aim to identify substructures in the graphs share similarities with their counterparts for binary tabular data. However, it is not enough to simply replace the subgraph patterns by their assigned codewords. The information about

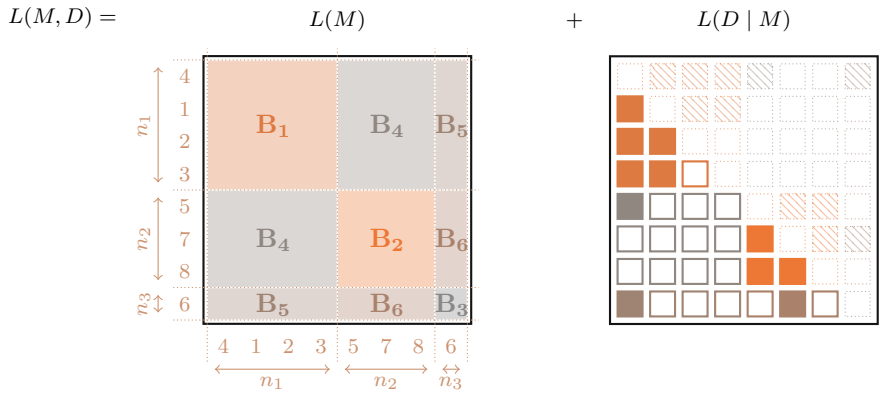


Fig. 7 Block-based strategy, example on the toy graph of Fig. 6. Ignoring labels and considering its adjacency matrix, the graph can be encoded in a very similar way as a binary tabular dataset (see Section 2.3 and Fig. 4). Furthermore, since the graph is undirected, and its matrix hence symmetric, it is enough to encode the lower triangular part of the matrix (depicted with solid lines and colour fill), from which the upper triangular part (depicted with dotted lines and hatch pattern) can be reconstructed. More intense shades of orange represent higher probabilities of ones within the corresponding block.

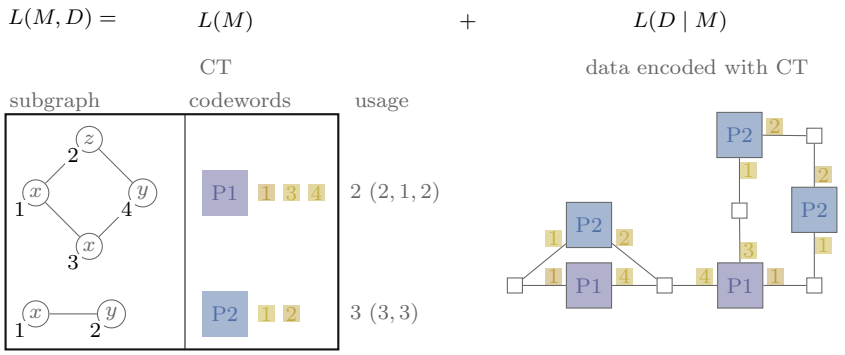


Fig. 8 Dictionary-based strategy, example on the toy graph of Fig. 6. The idea is similar to the one for binary tabular datasets (see Section 2.3 and Fig. 3). However, it is not enough to simply replace the subgraph patterns by their assigned codewords (depicted as blue blocks), the information about how the subgraphs are connected also needs to be encoded. Here, this is done through *ports* associated to the patterns and their assigned codewords (depicted as tan blocks).

how the subgraphs are connected also needs to be encoded, requiring more complex encoding schemes.

The survey by Liu et al. (2018a) covers graph summarisation methods in general, whereas Feng (2015) presents various information-theoretic graph mining methods, both of which include MDL-based methods for analysing graphs as a subset.

5.1 Grouping nodes into blocks

Rosvall and Bergstrom (2007) present an information-theoretic framework to identify community structure in networks by grouping nodes and propose to use the MDL principle to automatically select the number of groups in which to arrange the nodes.

Chakrabarti (2004) proposes to compress the adjacency matrix of a graph by grouping nodes into homogeneous blocks (see Fig. 7), with a top-down procedure to search for a good partition. Navlakha et al. (2008) similarly propose to build graph summaries by grouping nodes into supernodes, but with a bottom-up search procedure. A superedge linking two supernodes represents edges between all pairs of elementary nodes from either supernodes (hence a supernode with a loop represents a clique). When reconstructing the original graph, after expanding the supernodes and superedges, some corrections must be performed, to add and remove spurious edges. Navlakha et al. (2008) let the cost of encoding a graph equal the number of superedges and edge corrections, ignoring the cost of the assignment of nodes to supernodes.

Khan et al. (2015b) (also Khan, 2015; Khan et al., 2015a) work with essentially the same encoding, using locality-sensitive hashing (LSH) to identify candidates for merger, additionally considering node labels (Khan et al., 2014). Akoglu et al. (2012a) also aim at grouping nodes while taking into account node attributes.

A similar block summary approach for bipartite graphs is proposed by Feng et al. (2012), with a more complete encoding. Papadimitriou et al. (2008) also focus on bipartite graphs, but the obtained blocks are arranged into a hierarchy, while He et al. (2009) consider k-partite graphs.

In order to compress the adjacency matrix of an input graph more efficiently, He et al. (2011) look for nodes with similar connection patterns, corresponding to similar rows in the matrix, and encode the differences, possibly in a recursive manner. The approach is used to spot nodes with unusual connections patterns, that do not lend themselves to grouping.

LeFevre and Terzi (2010) propose a supernode summary involving superedge weights that represent the probability that an edge exists for each pair of nodes in the incident supernodes. In one variant of the problem, the MDL principle is used to choose the number k of supernodes that strikes the best balance between model complexity (k) and fit to the data (reconstruction error). Lee et al. (2020) also consider summarising graphs by grouping nodes together, but fix a maximum length for the description of the model, i.e. the hypernodes, and look for the summary that minimises the reconstruction error, measured as the length of the description of edge corrections.

Plant et al. (2020) use a MDL-inspired score to learn graph embeddings. That is, the aim is to project the nodes into a multi-dimensional space, so that the structure of the graph is preserved as much as possible and, more specifically, such that connected nodes are placed close to each other. Therefore, the distance between any pair of nodes in the embedding is used to compute a probability that the nodes are connected, which, in turn is used to encode

the presence or absence of the corresponding edge. Then, the quality of an embedding can be measured by how much it allows to compress the adjacency matrix.

5.2 Grouping nodes into blocks in dynamic graphs

Sun et al. (2007) introduce a block summary approach for dynamic graphs, extended to multiple dimensions or contexts by Jiang et al. (2016). Araujo et al. (2014b) also propose a block summary approach for dynamic graphs, which they later extend to multiple dimensions or contexts as represented by qualitative labels on the edges (Araujo et al., 2016).

5.3 Finding hyberbolic communities

Instead of looking for blocks of uniform density and motivated by the observation that node degrees in real-world networks often follow a power-law distribution, Araujo et al. (2014a) propose the model of *hyperbolic communities*. The name refers to the fact that when nodes in such communities are ordered by degree, edges in the adjacency matrix mostly end up below a hyperbola.

Kang and Faloutsos (2011) (also Lim et al., 2014) decompose the input graph into hubs and spokes, with superhubs connecting the hubs recursively, and introduce a cost to evaluate how well the decomposition allows to compress the graph. This type of decomposition is proposed as an alternative to a decomposition into cliques, referred to as "cavemen communities".

5.4 The Map Equation

Rosvall and Bergstrom (2008) propose a method to reveal the important connectivity structure of weighted directed graphs. The approach assigns codes to nodes in such a way that random walks over the graph can be described succinctly. Furthermore, nodes are partitioned into modules so that the codes for nodes are unique within each module but can be reused between modules. A walk over the graph can then be described using a combination of codes indicating transitions between modules and lists of the successive nodes encountered within each module. The resulting two-level summary of the graph maps its main structures and the connections between and within them, and the approach is therefore referred to as the *Map Equation* (Rosvall et al., 2009) or the *Infomap* algorithm.[4]

Later, refinements and extensions of the method were proposed, to study changes in the connectivity structure over time (Rosvall and Bergstrom, 2010), reveal multi-level hierarchical connectivity structure (Rosvall and Bergstrom, 2011), support overlaps between modules (Viamontes Esquivel and Rosvall,

[4] `https://www.mapequation.org/`

2011), among others (Bohlin et al., 2014; De Domenico et al., 2015; Edler et al., 2017a; Emmons and Mucha, 2019; Calatayud et al., 2019). In particular, the method has found application in the analysis of ecological communities (Edler et al., 2017b; Blanco et al., 2021; Rojas et al., 2021).

5.5 Identifying substructures

Cook and Holder (1994) (also Ketkar et al., 2005) propose the SUBDUE algorithm to mine substructures from graphs, possibly with labels, using the MDL principle. A substructure of the graph can be encoded and its occurrences in the graph be replaced by a single node. This can be done recursively, generating a hierarchical summary of the original graph. There are two shortcomings to the approach. First, replacing the substructure by a single node does not preserve the complete information about the connections to neighbours. Second, the matching of substructures is done in an approximate way, with an arbitrary fixed cost, rather than a proper encoding of the reconstruction errors (using the MDL principle for this evaluation is left for future work). Substructures are scored individually rather than in combination. The SUBDUE algorithm is used by Jonyer et al. (2004) for the induction of context-free grammars, and by Bloem (2013) in comparative experiments against practical data compression with the GZIP algorithm.

Bloem and de Rooij (2018) (also Bloem and de Rooij, 2020) propose to use the MDL principle when evaluating the statistical significance of the presence of substructures in a graph.

The VoG algorithm presented by Koutra et al. (2014) (also Koutra et al., 2015) allows to decompose the graph into basic primitives such as cliques, stars, and chains, which can overlap on nodes (but not on edges). Error corrections are then applied, to add and remove spurious edges. This can be seen as a global use of primitives. Liu et al. (2015) (also Liu et al., 2016) use the MDL principle to compare the ability of VoG and graph clustering methods to generate graph summaries. Liu et al. (2018b) build on VoG and address some of the shortcomings, such as the bias towards star structures, the inability to exploit edge overlaps, and the dependency on candidate order. Goebl et al. (2016) introduce a similar approach, with some of the same primitives, but prohibiting overlaps with the aim to make visualisation and interpretation easier. The approach presented by Bariatti et al. (2020a) removes the limitation to a predefined set of primitives and considers labelled graphs (see Fig. 8). The authors later upgraded the approach by generating candidates on-the-fly, thereby providing an anytime mining algorithm (Bariatti et al., 2021), and proposed a visualisation tool for the obtained graph patterns (Bariatti et al., 2020b). This work forms the basis of a doctoral dissertation (Bariatti, 2021).

The approach proposed by Coupette and Vreeken (2021) aims to highlight similarities and differences between graphs, and is akin in spirit to the DIFFNORM algorithm for transactional data (cf. Section 3.3). It looks for a common model consisting of basic primitives, like those used in VoG, such that

each graph can be reconstructed based on these primitives adjusted through parameters specific to the graph, as well as additional structures, where necessary.

Feng et al. (2013) exploit basic structures of graphs, like stars and triangles, to save on the encoding of the adjacency matrix. Such primitives assign a probability to the existence of an edge, which is used to encode it. Which primitive applies is determined in part based on structure information available so far, i.e. previously decoded. This can be seen as a local use of primitives.

Belth et al. (2020) learn relational rules which can be used to summarise knowledge graphs, involving typed edges and nodes.

5.6 Identifying substructures in dynamic graphs

Shah et al. (2015) (also Shah et al., 2017) extend the VoG approach to dynamic graphs. More specifically, they incorporate the temporal aspect of substructures appearing only at given time steps, across a range of contiguous time steps, periodically, or in a flickering fashion. Therefore, in addition to decomposing the graph into basic structures, one needs to indicate when these structures appear. The Mango algorithm by Saran and Vreeken (2019) also looks for predefined structures in a dynamic graph, aiming more specifically at tracking their evolution through time.

5.7 Finding pathways between nodes

Given a large graph, Akoglu et al. (2013) consider the problem of identifying a set of marked nodes. This can be done by listing the node identifiers or by navigating between nodes. The latter strategy requires to choose between the limited number of neighbours of each traversed node, rather than among all possible nodes in the graph, potentially leading to shorter descriptions. In particular, the problem formulated by Akoglu et al. (2013) consists in finding the best collection of trees spanning the marked nodes in the graph. The graph as such is not encoded, it is regarded as shared knowledge.

Prakash et al. (2014) similarly assume shared knowledge of the graph. The aim is then to transmit the starting points and spread of an epidemic through the graph over a sequence of time steps, assuming a "susceptible–infected" (SI) epidemic model.

6 Temporal data

In this section, we look at data where the attribute values come as a sequence, i.e. in a specific order. In particular, this order might correspond to time, in which case the data is called *temporal*. In some cases only the order matters, whereas in other cases absolute positions are associated to the values, such as timestamps in temporal data. In addition to time, spatial dimension(s) might

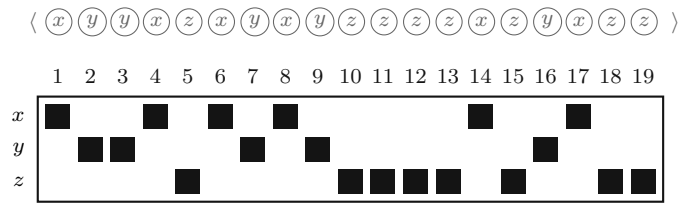


Fig. 9 A toy sequence, with nineteen consecutive occurrences over three distinct events x, y and z (top), also represented as a binary matrix (bottom) indicating which event (row) occurs at a given time step (column).

be associated to the values, resulting in *spatio-temporal* data. The terms *sequential data* and *sequence* are sometimes used to refer more narrowly to sequences of discrete attributes or items, which are typically called *events*. On the other hand, the term *timeseries* is generally used to refer to real-valued attributes sampled at regular or irregular time intervals. Text and genetic data (such as DNA or RNA sequences) fall into the former category. More specifically, such data generally comes in the form of *strings*, that is, as sequences of characters that represent occurrences of single items where the order is meaningful, not the positions. The data might consist of a single long sequence or of a database of multiple, typically shorter, sequences.

As with other types of data, most of the work on mining sequential data can be divided into two main tasks, namely segmentation and frequent pattern mining, corresponding to *block-based* and *dictionary-based* strategies, respectively (cf. Section 2.3). In segmentation problems (a.k.a. change point detection), the aim is to divide the input data into homogeneous blocks or segments, each associated to specific occurrence probabilities of the different events. On the other hand, in frequent pattern mining, the aim is to find recurrent substructures, which are commonly referred to as *episodes* and *motifs* when considering sequences and timeseries, respectively.

For illustrative purposes, we consider a toy sequence, shown in Fig. 9, and delineate approaches that follow either strategy. The example shown in Fig. 10 illustrates the *block-based* strategy and follows the work of Kiernan and Terzi (2008) (cf. Section 6.2), whereas the example shown in Fig. 11 illustrates the *dictionary-based* strategy and follows the work of Tatti and Vreeken (2012b) (cf. Section 6.5). While similar to their counterparts for binary tabular data, approaches for temporal data must account for the order that the special dimension of time imposes on the occurrences.

6.1 Finding haplotype blocks

A haplotype, or haploid genotype, is a group of alleles of different genes on a single chromosome, which are closely linked and typically inherited as a unit. Several works have been dedicated to the problem of finding haplotype block

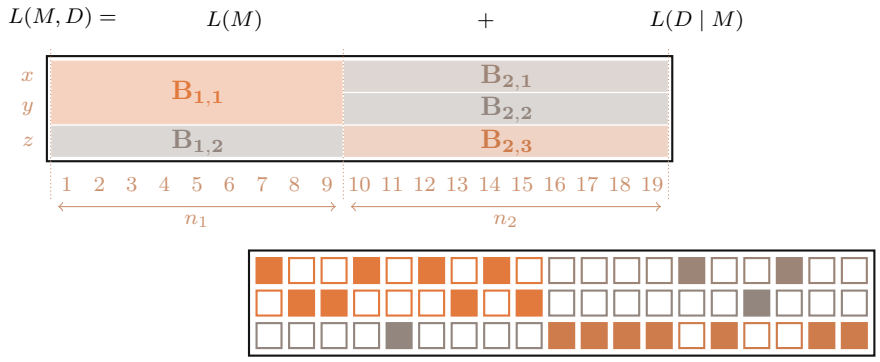


Fig. 10 Block-based strategy, example on the toy sequence of Fig. 9. The sequence can be encoded in a very similar way as a binary tabular dataset (see Section 2.3 and Fig. 4). In this case, however, the order of the columns corresponds to time and is therefore fixed, but the events (i.e. rows) can be arranged into different groups in the different time segments (i.e. column groups). More intense shades of orange represent higher probabilities of ones within the corresponding block.

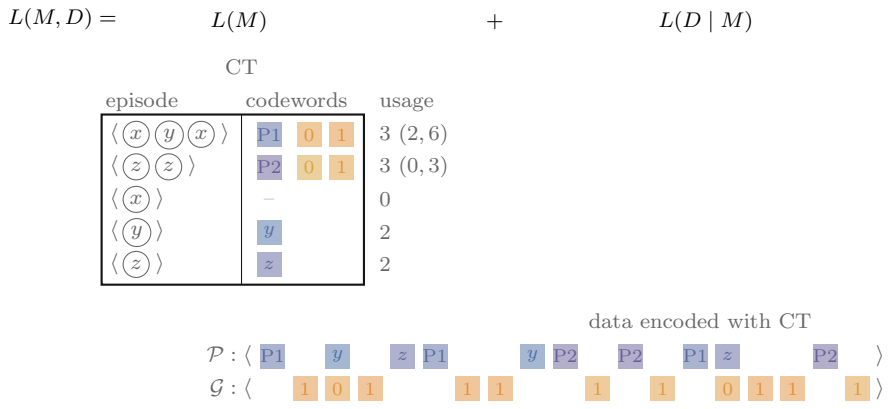


Fig. 11 Dictionary-based strategy, example on the toy sequence of Fig. 9. The idea is similar to the one for binary tabular datasets (see Section 2.3 and Fig. 3). Episode patterns are assigned codewords (depicted as blue blocks). The original sequence is reconstructed by reading codewords from the *pattern stream* ($\mathcal{P}$) and inserting the events of the corresponding episode into the sequence, in order. In addition, each multi-event episode in the code table is associated to an additional pair of codewords (depicted as bronze blocks). These codewords, read from the *gap stream* ($\mathcal{G}$), indicate whether to insert the next element from the current episode (1) or to leave a gap where to insert the next episode (0).

boundaries, i.e. identifying block structure in genetic sequences (Koivisto et al., 2002; Anderson and Novembre, 2003; Greenspan and Geiger, 2003; Mannila et al., 2003; Greenspan and Geiger, 2004). This requires jointly partitioning multiple aligned strings.

6.2 Segmenting sequences

Several approaches have also been developed for segmenting event sequences more in general.

The method introduced by Kiernan and Terzi (2008) partitions a sequence into time segments, then partitions the events of each segment into groups (see Fig. 10). The proposed algorithm is then extended to allow overlaps and gaps between segments (Kiernan and Terzi, 2009a) and a tool to visualise the obtained segmentation is proposed (Kiernan and Terzi, 2009b). Wang et al. (2010) further aim to model dependencies between segments.

The algorithm proposed by Lam et al. (2014a) partitions the alphabet into subsets, then separately encodes the sequence projected on each subset of symbols, as well as a sequence that maps each position to the corresponding subset. Chen et al. (2018) adopt a generic point of view on sequence segmentation, considering that the input data can be either univariate or multivariate, consist of categorical or real-valued variables, with no assumption on the underlying distribution. Gautrais et al. (2020) aim to segment a sequence in such a way that each segment contains a collection of recurrent "signature" events. In particular, they consider retail data and apply the approach to the sequences of transactions of individual customers, in order to analyse their shopping behaviour.

6.3 Segmenting timeseries

Hu et al. (2011) (also Hu et al., 2013, 2015) propose an approach to represent a timeseries with an Adaptive Piecewise Constant Approximation (APCA) and define the intrinsic cardinality of the data to be the number of distinct constant values in the approximation. In this approach, a timeseries is encoded by specifying the end-points and value of each segments, then listing reconstruction errors. Vespier et al. (2012) also decompose timeseries into segments, but consider components at multiple time-scales, modeled as piecewise constant or polynomial functions.

Matsubara et al. (2014) consider the problem of segmenting multivariate timeseries. A timeseries is modeled using a multi-level hidden Markov model (HMM), where high-level states represent regimes that contain lower level states. Wu et al. (2020) also consider a problem of timeseries segmentation, and model each segment with a Markov chain.

Rakthanmanon et al. (2011) (also Rakthanmanon et al., 2012) consider the problem of clustering sequential data, in particular such as arises from discretised timeseries where each distinct numerical value is mapped to a distinct symbol. The authors argue that not every value is of interest, because sequences tend to contain meaningless transitions, and that the MDL principle can help identify the segments of interest. Begum et al. (2013) (also Begum et al., 2014) use the clusters obtained with this approach for the task of semi-supervised classification.

6.4 Mining substrings

The algorithm devised by Evans et al. (2006) (also Markham et al., 2009) searches for the best set of substrings to encode an input string according to the proposed Optimal Symbol Compression Ratio (OSCR) (Evans et al., 2003). The algorithm, which has been applied primarily to analyse genetic sequences (Evans et al., 2007), is iterative, at each step picking the substring that compresses most and replacing it by a temporary code. Selected substrings can be recursive, in the sense that they contain previously selected substrings. In the end, the selected substrings are assigned codes using Huffman coding.

6.5 Mining episodes from sequences

Lam et al. (2012) propose to encode timestamped sequences with absolute positioning. That is, the positions of covered occurrences are listed separately for each singleton event or selected subsequence. A fixed-length code is used, so all elements (event or position) cost the same and, in particular, occurrences can appear arbitrarily far apart with no penalty. Follow-up work (Lam et al., 2014b) focuses on strings. The proposed algorithms have the same names (SeqKrimp and GoKrimp), but use a different encoding mechanism. Specifically, having constructed a dictionary mapping subsequences to codewords, each match of a selected subsequence is replaced by its associated codeword, followed by Elias codes indicating the gaps between occurrences of the successive events of the subsequence. For a given subsequence, a subroutine is proposed to find the matches with minimum gap cost. Lam et al. (2013) consider a similar problem in a streaming setting. The proposed encoding points back to the previous occurrence of the subsequence, with a flag to indicate when an extended subsequence should be recorded as new, that is, added to the dictionary.

The Sqs ("squeeze") algorithm of Tatti and Vreeken (2012b) follows a dictionary-based strategy and is similar to Krimp but for sequences (see Fig. 11). Each selected subsequence, or episode, is assigned a codeword representing it, as well as a pair of codewords representing gap (move to next position) and fill (insert event) operations. Gaps are allowed but not interleaving. In other words, gaps must be filled by singletons. This work is then extended in multiple ways, to take into account an ontology over the events, resulting in algorithm Nemo (Grosse and Vreeken, 2017) and by adding support for rich interleaving and choice of events in patterns, resulting in algorithm Squish (Bhattacharyya and Vreeken, 2017).

After focusing on the analysis of seismic data, aiming to cluster and compare seismograms represented as multiple aligned sequences (Bertens and Siebes, 2014), Bertens et al. (2016) consider multivariate event sequences more in general and propose algorithm Ditto, which can be seen as an extension of Sqs to handle multivariate patterns. The work constitutes the basis of a dissertation focused on detecting anomalies and mining multivariate event se-

quences, also in combination, i.e. employing Ditto to detect anomalies in such sequences (Bertens, 2017, Chapter 7). Hinrichs and Vreeken (2017) use compression and the Sqs algorithm to analyse the similarities between sequence databases in terms of occurring sequential patterns, focusing mostly on text data. The proposed algorithm, called SqsNorm, provides for sequential data the type of analysis that DiffNorm allows for transactional data (cf. Section 3.3).

The approach proposed by Wiegand et al. (2021) is clearly related to Sqs and Squish but aims to summarise entire complex event sequences, rather than capturing fragmentary behaviour. The models considered resemble Petri nets or finite state machines and specify conditional transitions between events. The data is then represented as a succession of instructions that, when fed through the model, allow to reconstruct the original event log. The authors later present a similar model called event-flow graph (Wiegand et al., 2022). Instead of pattern nodes, this model involves rules defined over attribute vectors associated to the sequences.

Cüppers and Vreeken (2020) aim to identify sequential patterns that reliably predict the impending occurrence of an event of interest. In other words, they look for a set of rules, but in sequential rather than transactional data (cf. Section 3.4). Along similar lines, Bourrand et al. (2021a) (also Bourrand et al., 2021b) aim to discover a compact set of sequential rules from a single long event sequence.

Fowkes and Sutton (2016) propose a generative probabilistic model of sequence databases. The authors discuss the connection between probabilistic modeling and description length, and compare their proposed algorithm, which is not based on the MDL principle, to Sqs and GoKrimp.

Ibrahim et al. (2016) consider sequences with timestamps and patterns that consist of subsequences with fixed inter-event times. The data is encoded by listing the patterns, along with their occurrences. More specifically, for each subsequence, the events and inter-event times are specified, as well as the timestamp of the first event of each occurrence. Mitra and Sastry (2019) propose a generative statistical model (a hidden Markov model, HMM) as justification for this encoding. Yan et al. (2018) look for patterns in a stream of sequential data where each event is associated to a timestamp, using a sliding window and maximum gap constraint.

6.6 Mining motifs from timeseries

Tanaka and Uehara (2003) (also Tanaka et al., 2005) look for motifs in timeseries, representing discretised values as integers using a fixed-length code.

Shokoohi-Yekta et al. (2015) consider the problem of extracting rules from timeseries, aiming to match shapes rather than the precise values. The proposed score is used to evaluate the consequent of candidate rules, allowing to compare consequents of different lengths. The score evaluates the compression gain resulting from specifying the motif once and then listing the errors for

each occurrence, instead of listing the actual values for each occurrence. This is applied to evaluate candidates individually, not as a set.

6.7 Mining periodic patterns

Exploiting regularities not only about what happens, that is, finding coordinated event occurrences, but also about when it happens, that is, finding consistent inter-occurrence time intervals, can allow to further compress the data.

In the context of "smart homes" and health monitoring, Heierman et al. (2004) look for periodically repeating events or sets of events in a sequence, with a MDL criterion to identify interesting candidates, which are then used to automatically construct a Markov model (HPOMDP) (also Heierman and Cook, 2003; Das and Cook, 2004; Youngblood et al., 2005). The work of Rashidi and Cook (2013) shares the same context and goal, further accounting for discontinuities in the repetitions and variations in the order of the events.

Galbrun et al. (2018) introduce patterns involving nested periodic recurrences of different events and a method for constructing them by combining simple cycles into increasingly complex and expressive patterns.

6.8 Trajectories

Phan et al. (2013) consider data that represent a collection of moving objects, each associated at each timestamp with a geospatial position. As a preprocessing step, the objects must be clustered based on proximity, separately for each timestamp. An object is allowed to belong to several clusters at any given timestamp. The goal is then to find a sequence of clusters (at most one per timestamp) having objects in common. The result is called a *swarm* and intended to represent objects moving together. When encoding the trajectory of an object (as a sequence of clusters) patterns can be used whole, or from/to an intermediate position.

Also considering spatio-temporal data but in a different scenario, Zhao et al. (2019) aim to mine frequent patterns in trajectories over a road network. Mapping the trajectories to the corresponding road segments effectively turns the problem into a frequent sequence mining problem. The MDL principle is used to formulate a problem of trajectory spatial compression, addressed using a dictionary-based strategy.

7 Discussion

Our goal here is to open a discussion on issues relevant to MDL-based methods for pattern mining. In particular, the design of the encoding is a crucial ingredient when developing such a method. We consider different questions that might be raised by the involved choices, regarding, in particular, conformity

and suitability with respect to the MDL principle. To illustrate the discussion, we point to various works listed in the previous sections.

7.1 Encoding in question

Alleged infractions to the MDL principle can be of different kinds and degrees of severity. They include cases where *(i)* the assignment of codewords ignores information theory, *(ii)* the proposed encoding is not functional due to some information missing, and *(iii)* the proposed encoding clearly cannot achieve a good compression due to the presence of unnecessary unjustified terms.

Several methods assign the same, typically unit, cost to all encoded elements (which might be items, nodes, edges, events, timestamps, etc. depending on the case, or even entire patterns), so that the description length is simply the number of encoded elements (see for instance Navlakha et al. (2008) in Section 5.1, and Phan et al. (2013); Zhao et al. (2019) in Section 6.8). Some authors motivate this choice by the need to avoid penalising large values, or to circumvent other encoding issues (see for instance Ibrahim et al. (2016); Lam et al. (2012) in Section 6.5). In the method proposed by Yan et al. (2018) (cf. Section 6.5), for instance, the cost of a pattern is first defined to be the number of characters used to represent it, that is, the number of events plus the number of timestamps. In a second version of the method, this cost is then defined to be equal to one for all patterns, reportedly to avoid bias against patterns involving more events.

Transmitting the dataset through a binary communication channel as efficiently as possible is a thought experiment of sorts that motivates the score. If short codewords are assigned to specific elements because they are deemed more valuable and useful, then other elements will have to be assigned longer codewords, because not everything can be transmitted cheaply. One can think of it as the fundamental limits of information theory, through this compression scenario, forcing the designer of the method to make choices as to what he considers important and interesting. One might argue that using unit costs corresponds to using a fixed-length code and rescaling everything for convenience. This indeed simplifies the design of the encoding, as it avoids making decisions, and in this sense short-circuits the principle.

Small coding elements, such as for example required to delimit patterns in the code table, are often omitted. This is sometimes done deliberately, putting forth, in particular, the use of a pre-defined framework to be filled with the relevant values, which is common to all models and can therefore be ignored (see for instance Vreeken et al. (2011) in Section 3, and van Leeuwen and Galbrun (2015) in Section 3.4), but is sometimes left unexplained and might seem accidental. In the approach proposed by Lam et al. (2014b) (cf. Section 6.5), it is unclear how the receiver knows where the codewords end when decoding the dictionary. On the other hand, Tanaka and Uehara (2003) (cf. Section 6.6) use a fixed-length code to encode values from a set, but the

number of distinct values, which varies for different models, is not transmitted, so that the receiver cannot deduce the codeword length, and hence cannot decode the message.

More substantial pieces might also be missing. For instance, the encodings proposed by Lam et al. (2014a) (cf. Section 6.2) and by Hu et al. (2011) (cf. Section 6.3) do not account for the transmission of the assignment of symbols to subsets and of the mapping of offset values to codewords for the corrections, respectively, which are needed to reconstruct the data.

Explanations about the encoding are sometimes kept at the level of intuitions, and the details provided can be insufficient to properly understand how it works (see for instance Khan et al. (2015b) in Section 5.1, Matsubara et al. (2014) in Section 6.3, Heierman et al. (2004); Rashidi and Cook (2013) in Section 6.7, and Phan et al. (2013) in Section 6.8).

Arguably, ensuring decodability would in some cases require only minor modifications of the encoding scheme, and would likely have no major impact on the results. Furthermore, how much effort should be spent ensuring that the proposed encoding works is debatable, since it will never be used in practice.

The choice of encoding can sometimes seem sub-optimal, ill-suited or introduce undesirable bias. For instance, Navlakha et al. (2008) (cf. Section 5.1) list edge corrections that should be applied to the reconstructed graph, indicating for each one the sign of the correction. It seems, however that this information is unnecessary, as it can be inferred from the reconstructed graph, by checking whether the edge is present (must be deleted) or absent (must be added). Hu et al. (2011) (cf. Section 6.3) encode a list of value corrections using Huffman coding, meaning that having few distinct but recurrent error values is rewarded, not necessarily small ones. Using a universal code for the corrections would instead encourage small error values, which might be more intuitive. In any case, it is advisable to lay bare and motivate the potential biases introduced by the choice of encoding, whenever possible.

Considering a sequence, Lam et al. (2013) (cf. Section 6.5) encode the occurrence of an event or subsequence by pointing back to the position of the first occurrence. Pointing back, instead, to the position of the last encountered occurrence would require to encode smaller values and might lead to savings. Keeping track of the order in which the patterns were last encountered and referring to the position in that list, so that repetitions of the same pattern do not fill up the list, is another alternative. There are often different ways to achieve the same purpose, not necessarily with a clear overall best choice. In addition to pointing back to previous occurrences, Lam et al. (2013) maintain a dictionary of patterns. It is unclear whether the dictionary is actually needed for the encoding, or is primarily used to recover the encountered patterns.

What is part of the encoding of the model and what is part of the encoding of the data given the model is sometimes not entirely obvious. For example, the algorithm of Lam et al. (2014a) (cf. Section 6.2) encodes a sequence by partitioning the alphabet and considering separately the subsequences over each subset of symbols. The authors present the term that corresponds to

the assignment of positions to subsets as part of the encoding of the model. Debatably, it can be considered instead as part of the encoding of the data given the model, while the assignment of symbols to subsets, which is ignored, would belong to the encoding of the model. Besides, encodings often actually consist of three terms, *(i)* a description of the set of patterns (the model), *(ii)* information to reconstruct the data using these patterns, and *(iii)* a list of corrections to apply to the reconstructed data in order to recover the original data, with the latter two together representing the data given the model.

7.2 Code of choice

Prequential plug-in codes, and refined codes more in general, provide means to avoid unwanted bias arising from arbitrary choices in the encoding (cf. Section 2).

For instance, Budhathoki and Vreeken (2015) use prequential coding for the itemset occurrences in the DiffNorm algorithm (cf. Section 3.3). The choice is especially relevant in this scenario where the goal is to contrast the itemset make-up of different datasets, and not to inspect the usage of itemsets in a particular dataset. Bhattacharyya and Vreeken (2017) as well as Wiegand et al. (2021) use prequential coding for the streams that contain information about pattern occurrences (cf. Section 6.5). Other recent works (Faas and van Leeuwen, 2020; Makhalova et al., 2020; Bloem and de Rooij, 2020, cf. Sections 4.2, 4.6 and 5.5, respectively) also use prequential coding, while Bertens et al. (2016); Hinrichs and Vreeken (2017) (cf. Section 6.5) both explicitly suggest upgrading the current encoding with a prequential code, as a direction for future work. Going further, Proença et al. (2021a) improved on their earlier work (Proença and van Leeuwen, 2020a) (cf. Section 3.4) by replacing prequential coding, which is only asymptotically optimal, with normalised maximum likelihood (NML), which is optimal for fixed sample sizes, employing similar techniques as Kontkanen and Myllymäki (2007) (cf. Section 4.6).

However, modern Bayesian and NML codes can be challenging to compute, or even downright infeasible. Furthermore, one-part codes can be less intuitive than two-part codes, and do not provide as direct an access to information about pattern usage. For instance, Mampaey and Vreeken (2010) (cf. Section 4.1) compare two encodings, with and without prequential coding, and, obtaining similar results, choose to proceed with the latter as it is more intuitive. All in all, modern refined codes have improved theoretical properties, but using them to build better methods comes with some challenges.

7.3 The letter or the spirit

Some approaches use the MDL principle to score and compare individual candidate patterns, rather than evaluating them in combination (see for instance

Cook and Holder (1994) in Section 5.5, Shokoohi-Yekta et al. (2015) in Section 6.6, as well as Heierman et al. (2004); Rashidi and Cook (2013) in Section 6.7).

Considering a two-view dataset, i.e. a dataset consisting of two tables, the approach proposed by van Leeuwen and Galbrun (2015) assumes knowledge of one table to encode the other, and vice versa. Arguably, this approach does not correspond to a practical encoding, like other MDL-based approaches, but also not to a realistic compression scenario, yet it serves as a reasonable motivation for the proposed score.

The proposed score might actually be entirely ad-hoc, in the sense that it does not correspond to the length of an encoding that could be used to represent the data (see for instance Makhalova et al. (2019a) in Section 4.6). One might reasonably devise and justify an evaluation measure suited to the problem at hand, but labelling it as following the MDL principle is arbitrary and inappropriate, short of an explanation of how this corresponds to encoding, and can only lead to confusion.

Authors sometimes approach the topic with caution and include disclaimers stating that their proposed methods are inspired by or in the spirit of the MDL principle (see for instance Shokoohi-Yekta et al. (2015) in Section 6.6). This can be seen as a way to allow oneself to take some liberties with the principle, indeed considering it as a source of inspiration rather than as law, but also as a way to preventively fend off criticism and accusations of heresy. There is indeed a range of opinions about how closely one must conform to the MDL principle and to information theory.

7.4 Making comparisons

How to make meaningful comparisons between compression-based scores and between corresponding results requires careful consideration. For instance, one might ponder whether the compression achieved for a dataset is an indication of how much structure is present in it, or at least how much could be detected, and to what extent it can serve as a measure of the performance of the algorithm.

Does it make sense to compare the length of a dataset encoded with the proposed scheme to the original unencoded data? And is it a problem if the latter is shorter? Keeping in mind that compression is used as a tool for comparing models, rather than for practical purposes, we answer both questions in the negative. The compression achieved with the simplest model, be it the code table containing only elementary patterns such as the singleton itemsets, known as the *standard code table* in Krimp (Vreeken et al. (2011) in Section 3), for dictionary-based approaches (cf. Fig. 3*(i)*) or the single-block model for block-based approaches (cf. Fig. 4*(i)*), is often considered as a basis for comparison. The ratio of the compression achieved with a considered model to the compression achieved with the elementary model, known as the *compression ratio*, is then computed and used to compare different models, with

lower compression ratios corresponding to better models. This is a way to normalise the scores and allow more meaningful comparisons and evaluations.

A direct comparison of the raw description lengths, in terms of numbers of bits, of the same data encoded with different methods is typically not meaningful. For instance, it does not make sense to compare the description lengths reported in Fig. 3 to those reported in Fig. 4. Comparing compression ratios across different methods is not really meaningful either in general. Indeed, an easy way to win this contest would be to design an artificial encoding that penalises very heavily the use of elementary patterns. If the different methods handle compatible pattern languages, comparing the compression ratios achieved when considering as model, in turn, the set of patterns selected by each method and applying either encoding can be of interest, and might shed some light on the respective biases of the methods. If the pattern languages are not compatible, then no quantitative comparison can be devised easily and great care must be taken to choose suitable encodings. Qualitative evaluations of obtained patterns are valuable, despite being subjective and domain dependent. In the end, finding a good set of interesting and interpretable patterns is what matters.

7.5 Beyond mining patterns with MDL

Finally, we highlight two directions of research that do not fall strictly within the realm of MDL-based pattern mining methods, yet are clearly related, constitute recently active and fruitful research topics, and might therefore be of interest to the reader.

The first direction consists of studies of correlation and causality that build on algorithmic information theory in general and, for a few of them, on MDL-based pattern mining techniques more in particular. For instance, Budhathoki and Vreeken (2017a) propose two algorithmic correlation measures and present practical instanciations based on the MDL principle, using the Slim and Pack algorithms (cf. Sections 3.2 and 4.1, respectively). They also propose to use a MDL score to infer the direction of causality between pairs of discrete variables (Budhathoki and Vreeken, 2017b).

The second direction consists in the development of pattern mining methods relying on a different modeling approach, also grounded in information theory, namely on maximum entropy modeling. In particular, De Bie et al. (2010) introduced a framework for data mining based on maximum entropy modeling, sometimes referred to as the *FORSIED* framework, for *Formalising Subjective Interestingness*, from which models are derived for different types of assumptions, data and patterns (including for instance Kontonasios and De Bie, 2012; Kontonasios et al., 2013; van Leeuwen et al., 2016; Adriaens et al., 2019; Puolamäki et al., 2020).

8 Conclusion

After giving an outline of relevant concepts from information theory and coding, we reviewed MDL-based methods for mining various types of data and patterns. In particular, we focused on aspects related to the design of an encoding scheme, rather than on algorithmic issues for instance, since the former constitutes the most distinctive ingredient of MDL methodologies, but also a major stumbling block and source of contention. We pointed out two main strategies that underpin the majority of approaches and that can be used to categorise them. Namely, we distinguished dictionary-based approaches from block-based approaches. Then, we considered some discussion points pertaining to the use of MDL in pattern mining, and highlighted related problems that constitute promising directions for future research. Indeed, there is still room for further development in mining patterns with MDL-inspired methods, and beyond.

References

Adriaens F, Lijffijt J, De Bie T (2019) Subjectively interesting connecting trees and forests. Data Mining and Knowledge Discovery 33(4):1088–1124, DOI 10.1007/s10618-019-00627-1

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, VLDB'94, Morgan Kaufmann, pp 487–499

Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. ACM SIGMOD Record 22(2):207–216, DOI 10.1145/170036.170072

Akoglu L, Tong H, Meeder B, Faloutsos C (2012a) PICS: Parameter-free identification of cohesive subgroups in large attributed graphs. In: Proceedings of the 2012 SIAM International Conference on Data Mining, SDM'12, SIAM, pp 439–450, DOI 10.1137/1.9781611972825.38

Akoglu L, Tong H, Vreeken J, Faloutsos C (2012b) Fast and reliable anomaly detection in categorical data. In: Proceedings of the 21st ACM international conference on Information and knowledge management, CIKM'12, ACM, pp 415–424, DOI 10.1145/2396761.2396816

Akoglu L, Chau DH, Vreeken J, Tatti N, Tong H, Faloutsos C (2013) Mining connection pathways for marked nodes in large graphs. In: Proceedings of the 2013 SIAM International Conference on Data Mining, SDM'13, SIAM, pp 37–45, DOI 10.1137/1.9781611972832.5

Anderson EC, Novembre J (2003) Finding haplotype block boundaries by using the minimum-description-length principle. American Journal of Human Genetics 73(2):336–354

Aoga JOR, Guns T, Nijssen S, Schaus P (2018) Finding probabilistic rule lists using the minimum description length principle. In: Proceedings of the

International Conference on Discovery Science, DS'18, Springer, pp 66–82, DOI 10.1007/978-3-030-01771-2_5

Araujo M, Günnemann S, Mateos G, Faloutsos C (2014a) Beyond blocks: Hyperbolic community detection. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'14, Springer, pp 50–65, DOI 10.1007/978-3-662-44848-9_4

Araujo M, Papadimitriou S, Günnemann S, Faloutsos C, Basu P, Swami A, Papalexakis EE, Koutra D (2014b) Com2: Fast automatic discovery of temporal ('comet') communities. In: Proceedings of 18th Pacific-Asia Conference on the Advances in Knowledge Discovery and Data Mining, PAKDD'14, Springer, pp 271–283, DOI 10.1007/978-3-319-06605-9_23

Araujo M, Günnemann S, Papadimitriou S, Faloutsos C, Basu P, Swami A, Papalexakis EE, Koutra D (2016) Discovery of "comet" communities in temporal and labeled graphs COM^2. Knowledge and Information Systems 46(3):657–677, DOI 10.1007/s10115-015-0847-2

Asadi B, Varadharajan V (2019a) An MDL-based classifier for transactional datasets with application in malware detection. arXiv:1910.03751

Asadi B, Varadharajan V (2019b) Towards a robust classifier: An MDL-based method for generating adversarial examples. arXiv:1912.05945

Bariatti F (2021) Mining tractable sets of graph patterns with the minimum description length principle. Phd thesis, Université de Rennes 1, URL `https://hal.inria.fr/tel-03523742`

Bariatti F, Cellier P, Ferré S (2020a) GraphMDL: Graph pattern selection based on minimum description length. In: Proceedings of the 18th International Symposium on Advances in Intelligent Data Analysis, IDA'20, Springer, pp 54–66, DOI 10.1007/978-3-030-44584-3_5

Bariatti F, Cellier P, Ferré S (2020b) GraphMDL visualizer: Interactive visualization of graph patterns. In: Proceedings of the Graph Embedding and Mining Workshop GEM@ECML/PKDD'20, URL `https://hal.inria.fr/hal-03142207`

Bariatti F, Cellier P, Ferré S (2021) GraphMDL+: interleaving the generation and MDL-based selection of graph patterns. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing, SAC'21, ACM, pp 355–363, DOI 10.1145/3412841.3441917

Bastide Y, Pasquier N, Taouil R, Stumme G, Lakhal L (2000) Mining minimal non-redundant association rules using frequent closed itemsets. In: Proceedings of the First International Conference on Computational Logic, CL'00, Springer, pp 972–986

Begum N, Hu B, Rakthanmanon T, Keogh E (2013) Towards a minimum description length based stopping criterion for semi-supervised time series classification. In: Proceedings of the 14th IEEE International Conference on Information Reuse Integration, IRI'13, IEEE Computer Society, pp 333–340, DOI 10.1109/IRI.2013.6642490

Begum N, Hu B, Rakthanmanon T, Keogh E (2014) A minimum description length technique for semi-supervised time series classification. Integration

of Reusable Systems pp 171–192, DOI 10.1007/978-3-319-04717-1_8
Belth C, Zheng X, Vreeken J, Koutra D (2020) What is normal, what is strange, and what is missing in a knowledge graph: Unified characterization via inductive summarization. In: Proceedings of The Web Conference, WWW'20, ACM, pp 1115–1126, DOI 10.1145/3366423.3380189
Bertens R (2017) Insight in information : from abstract to anomaly. Phd thesis, Universiteit Utrecht
Bertens R, Siebes A (2014) Characterising seismic data. In: Proceedings of the 2014 SIAM International Conference on Data Mining, SDM'14, SIAM, pp 884–892, DOI 10.1137/1.9781611973440.101
Bertens R, Vreeken J, Siebes A (2015) Beauty and brains: Detecting anomalous pattern co-occurrences. arXiv:1512.07048
Bertens R, Vreeken J, Siebes A (2016) Keeping it short and simple: Summarising complex event sequences with multivariate patterns. In: Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'16, ACM
Bertens R, Vreeken J, Siebes A (2017) Efficiently discovering unexpected pattern-co-occurrences. In: Proceedings of the 2017 SIAM International Conference on Data Mining, SDM'17, SIAM, pp 126–134, DOI 10.1137/1.9781611974973.15
Bhattacharyya A, Vreeken J (2017) Efficiently summarising event sequences with rich interleaving patterns. In: Proceedings of the 2017 SIAM International Conference on Data Mining, SDM'17, SIAM
Blanco F, Calatayud J, Martín-Perea DM, Domingo MS, Menéndez I, Müller J, Fernández MH, Cantalapiedra JL (2021) Punctuated ecological equilibrium in mammal communities over evolutionary time scales. Science 372(6539):300–303, DOI 10.1126/science.abd5110
Bloem P (2013) Compression-based inference on graph data. In: Proceedings of the 22nd annual Belgian-Dutch Conference on Machine Learning, BENELEARN'13
Bloem P, de Rooij S (2018) A tutorial on MDL hypothesis testing for graph analysis. arXiv:1810.13163
Bloem P, de Rooij S (2020) Large-scale network motif analysis using compression. Data Mining and Knowledge Discovery 34(5):1421–1453, DOI 10.1007/s10618-020-00691-y
Bobed C, Maillot P, Cellier P, Ferré S (2019) Data-driven assessment of structural evolution of RDF graphs. Semantic Web – Interoperability, Usability, Applicability
Bohlin L, Edler D, Lancichinetti A, Rosvall M (2014) Community detection and visualization of networks with the map equation framework. In: Ding Y, Rousseau R, Wolfram D (eds) Measuring Scholarly Impact: Methods and Practice, Springer International Publishing, pp 3–34
Boley M, Lucchese C, Paurat D, Gärtner T (2011) Direct local pattern sampling by efficient two-step random procedures. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'11, ACM, pp 582–590, DOI 10.1145/2020408.2020500

Boley M, Mampaey M, Kang B, Tokmakov P, Wrobel S (2013) One click mining: Interactive local pattern discovery through implicit preference and performance learning. In: Proceedings of the Workshop on Interactive Data Exploration and Analytics, IDEA @KDD'13, ACM, pp 27–35, DOI 10.1145/2501511.2501517

Bonchi F, van Leeuwen M, Ukkonen A (2011) Characterizing uncertain data using compression. In: Proceedings of the 2011 SIAM International Conference on Data Mining, SDM'11, SIAM, pp 534–545

Bourrand E, Galárraga L, Galbrun E, Fromont E, Termier A (2021a) Discovering useful compact sets of sequential rules in a long sequence. In: Proceedings of the 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence, ICTAI'21, IEEE Computer Society, pp 1295–1299, DOI 10.1109/ICTAI52525.2021.00204

Bourrand E, Galárraga L, Galbrun E, Fromont E, Termier A (2021b) Discovering useful compact sets of sequential rules in a long sequence. arXiv:2109.07519

Budhathoki K, Vreeken J (2015) The difference and the norm – characterising similarities and differences between databases. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'15, Springer, vol 9285, pp 206–223, DOI 10.1007/978-3-319-23525-7_13

Budhathoki K, Vreeken J (2017a) Correlation by compression. In: Proceedings of the 2017 SIAM International Conference on Data Mining, SDM'17, SIAM, pp 525–533, DOI 10.1137/1.9781611974973.59

Budhathoki K, Vreeken J (2017b) MDL for causal inference on discrete data. In: Proceedings of the 17th IEEE International Conference on Data Mining, ICDM'17, IEEE Computer Society, pp 751–756, DOI 10.1109/ICDM.2017.87

Calatayud J, Bernardo-Madrid R, Neuman M, Rojas A, Rosvall M (2019) Exploring the solution landscape enables more reliable network community detection. Physical Review E 100(5):052308, DOI 10.1103/PhysRevE.100.052308

Chakrabarti D (2004) AutoPart: Parameter-free graph partitioning and outlier detection. In: Proceedings of the European Conference on Knowledge Discovery in Databases, PKDD'04, Springer, pp 112–124, DOI 10.1007/978-3-540-30116-5_13

Chakrabarti D, Papadimitriou S, Modha DS, Faloutsos C (2004) Fully automatic cross-associations. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'04, ACM, pp 79–88, DOI 10.1145/1014052.1014064

Chen L, Amiri SE, Prakash BA (2018) Automatic segmentation of data sequences. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI'18, Association for the Advancement of Artificial Intelligence

Cook DJ, Holder LB (1994) Substructure discovery using minimum description length and background knowledge. Journal of Artificial Intelligence Research

1(1):231–255
Coupette C, Vreeken J (2021) Graph similarity description: How are these graphs similar? In: Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'21, ACM
Cover TM, Thomas JA (2012) Elements of information theory. John Wiley & Sons
Cüppers J, Vreeken J (2020) Just wait for it... mining sequential patterns with reliable prediction delays. In: Proceedings of the 20th IEEE International Conference on Data Mining, ICDM'20, IEEE Computer Society
Das SK, Cook DJ (2004) Health monitoring in an agent-based smart home. In: Proceedings of the International Conference on Smart Homes and Health Telematics, ICOST'04, IOS Press, pp 3–14
De Bie T, Kontonasios KN, Spyropoulou E (2010) A framework for mining interesting pattern sets. SIGKDD Explorations (and Proceedings of the ACM SIGKDD Workshop on Useful Patterns, UP'10) 12(2):92–100
De Domenico M, Lancichinetti A, Arenas A, Rosvall M (2015) Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. Physical Review X 5(1):11027, DOI 10.1103/PhysRevX.5.011027
De Raedt L, Zimmermann A (2007) Constraint-based pattern set mining. In: Proceedings of the 2007 SIAM International Conference on Data Mining, SDM'07, SIAM, pp 237–248, DOI 10.1137/1.9781611972771.22
Edler D, Bohlin L, Rosvall M (2017a) Mapping higher-order network flows in memory and multilayer networks with infomap. Algorithms 10(4):112, DOI 10.3390/a10040112
Edler D, Guedes T, Zizka A, Rosvall M, Antonelli A (2017b) Infomap bioregions: Interactive mapping of biogeographical regions from species distributions. Systematic Biology 66(2):197–204, DOI 10.1093/sysbio/syw087
Emmons S, Mucha PJ (2019) Map equation with metadata: Varying the role of attributes in community detection. Physical Review E 100(2):022301, DOI 10.1103/PhysRevE.100.022301
Evans S, Saulnier G, Bush SF (2003) A new universal two part code for estimation of string kolmogorov complexity and algorithmic minimum sufficient statistic. In: Proceedings of the DIMACS Workshop on Complexity and Inference
Evans S, Markham TS, Torres A, Kourtidis A, Conklin D (2006) An improved minimum description length learning algorithm for nucleotide sequence analysis. In: Proceedings of the 2006 Fortieth Asilomar Conference on Signals, Systems and Computers, ACSSC'06, pp 1843–1850, DOI 10.1109/ACSSC.2006.355081
Evans S, Kourtidis A, Markham TS, Miller J, Conklin DS, Torres AS (2007) MicroRNA target detection and analysis for genes related to breast cancer using MDLcompress. EURASIP Journal on Bioinformatics and Systems Biology 2007(1):43670, DOI 10.1186/1687-4153-2007-43670
Faas M, van Leeuwen M (2020) Vouw: Geometric pattern mining using the MDL principle. In: Proceedings of the 18th International Symposium on

Advances in Intelligent Data Analysis, IDA'20, Springer, pp 158–170, DOI 10.1007/978-3-030-44584-3_13

Feng J (2015) Information-theoretic graph mining. Phd thesis, Ludwig-Maximilians-Universität München

Feng J, He X, Konte B, Böhm C, Plant C (2012) Summarization-based mining bipartite graphs. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'12, ACM, pp 1249–1257, DOI 10.1145/2339530.2339725

Feng J, He X, Hubig N, Böhm C, Plant C (2013) Compression-based graph mining exploiting structure primitives. In: Proceedings of the 13th IEEE International Conference on Data Mining, ICDM'13, IEEE Computer Society, pp 181–190, DOI 10.1109/ICDM.2013.56

Fischer J, Vreeken J (2019) Sets of robust rules, and how to find them. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'19, ACM, pp 38–54, DOI 10.1007/978-3-030-46150-8_3

Fischer J, Vreeken J (2020) Discovering succinct pattern sets expressing co-occurrence and mutual exclusivity. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'19, ACM

Fischer J, Oláh A, Vreeken J (2021) What's in the box? explaining neural networks with robust rules. In: Proceedings of the 38th International Conference on Machine Learning, ICML'21

Fowkes J, Sutton C (2016) A subsequence interleaving model for sequential pattern mining. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'16, ACM, pp 835–844

Galbrun E, Cellier P, Tatti N, Termier A, Crémilleux B (2018) Mining periodic patterns with a MDL criterion. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'18, pp 535–551

Gallo A, De Bie T, Cristianini N (2007) MINI: Mining informative non-redundant itemsets. In: Proceedings of the European Conference on Knowledge Discovery in Databases, PKDD'07, Springer, pp 438–445, DOI 10.1007/978-3-540-74976-9_44

Gautrais C, Cellier P, van Leeuwen M, Termier A (2020) Widening for MDL-based retail signature discovery. In: Proceedings of the 18th International Symposium on Advances in Intelligent Data Analysis, IDA'20, Springer, pp 197–209, DOI 10.1007/978-3-030-44584-3_16

Geng L, Hamilton HJ (2006) Interestingness measures for data mining: A survey. ACM Computing Surveys 38(3), DOI 10.1145/1132960.1132963

Gionis A, Mannila H, Mielikäinen T, Tsaparas P (2007) Assessing data mining results via swap randomization. ACM Transactions on Knowledge Discovery from Data 1(3):14–es, DOI 10.1145/1297332.1297338

Goebl S, Tonch A, Böhm C, Plant C (2016) MeGS: Partitioning meaningful subgraph structures using minimum description length. In: Proceedings of

the 16th IEEE International Conference on Data Mining, ICDM'16, IEEE Computer Society, pp 889–894, DOI 10.1109/ICDM.2016.0108

Greenspan G, Geiger D (2003) Model-based inference of haplotype block variation. In: Proceedings of the seventh annual international conference on Research in computational molecular biology, RECOMB'03, ACM, pp 131–137, DOI 10.1145/640075.640092

Greenspan G, Geiger D (2004) Model-based inference of haplotype block variation. Journal of Computational Biology 11(2):493–504, DOI 10.1089/1066527041410300

Grosse K, Vreeken J (2017) Summarising event sequences using serial episodes and an ontology. In: Proceedings of the Workshop on Interactions between Data Mining and Natural Language Processing @ECML/PKDD'17

Grünwald PD (2007) The Minimum Description Length Principle. MIT Press

Guns T, Nijssen S, De Raedt L (2011) Itemset mining: A constraint programming perspective. Artificial Intelligence 175(12):1951–1983

Guns T, Nijssen S, De Raedt L (2013) k-pattern set mining under constraints. IEEE Transactions on Knowledge and Data Engineering 25(2):402–418, DOI 10.1109/TKDE.2011.204

Hämäläinen W, Webb GI (2018) A tutorial on statistically sound pattern discovery. Data Mining and Knowledge Discovery DOI 10.1007/s10618-018-0590-x

Hanhijärvi S, Ojala M, Vuokko N, Puolamäki K, Tatti N, Mannila H (2009) Tell me something i don't know: Randomization strategies for iterative data mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'09, ACM, pp 379–388, DOI 10.1145/1557019.1557065

He J, Tong H, Papadimitriou S, Eliassi-Rad T, Faloutsos C, Carbonell J (2009) PaCK: Scalable parameter-free clustering on k-partite graphs. In: Proceedings of the 2006 SIAM International Conference on Data Mining, SDM'09, SIAM, pp 1278–1287

He X, Feng J, Plant C (2011) Automatically spotting information-rich nodes in graphs. In: Proceedings of the 11th IEEE International Conference on Data Mining Workshops, ICDMW'11, IEEE Computer Society, pp 941–948, DOI 10.1109/ICDMW.2011.37

He X, Feng J, Konte B, Mai ST, Plant C (2014) Relevant overlapping subspace clusters on categorical data. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'14, ACM, pp 213–222, DOI 10.1145/2623330.2623652

Heierman EO, Cook DJ (2003) Improving home automation by discovering regularly occurring device usage patterns. In: Proceedings of the 3rd IEEE International Conference on Data Mining, ICDM'03, IEEE Computer Society, pp 537–540, DOI 10.1109/ICDM.2003.1250971

Heierman EO, Youngblood GM, Cook DJ (2004) Mining temporal sequences to discover interesting patterns. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'19, ACM

Heikinheimo H, Siebes A, Vreeken J, Mannila H (2009) Low-entropy set selection. In: Proceedings of the 2009 SIAM International Conference on Data Mining, SDM'09, SIAM, pp 569–580, DOI 10.1137/1.9781611972795.49

Hess S, Piatkowski N, Morik K (2014) SHrimp: Descriptive patterns in a tree. In: Proceedings of the LWA (Lernen, Wissen, Adaption) 2014 Workshops: KDML, IR, FGWM

Hess S, Morik K, Piatkowski N (2017) The PRIMPING routine – tiling through proximal alternating linearized minimization. Data Mining and Knowledge Discovery 31(4):1090–1131, DOI 10.1007/s10618-017-0508-z

Hinrichs F, Vreeken J (2017) Characterising the difference and the norm between sequence databases. In: Proceedings of the Workshop on Interactions between Data Mining and Natural Language Processing @ECML/PKDD'17

Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E (2011) Discovering the intrinsic cardinality and dimensionality of time series using MDL. In: Proceedings of the 11th IEEE International Conference on Data Mining, ICDM'11, IEEE Computer Society, pp 1086–1091, DOI 10.1109/ICDM.2011.54

Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E (2013) Towards discovering the intrinsic cardinality and dimensionality of time series using MDL. In: Proceedings of the Ray Solomonoff 85th Memorial Conference, Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence, Springer, pp 184–197, DOI 10.1007/978-3-642-44958-1_14

Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E (2015) Using the minimum description length to discover the intrinsic cardinality and dimensionality of time series. Data Mining and Knowledge Discovery 29(2):358–399, DOI 10.1007/s10618-014-0345-2

Ibrahim A, Sastry S, Sastry PS (2016) Discovering compressing serial episodes from event sequences. Knowledge and Information Systems 47(2):405–432, DOI 10.1007/s10115-015-0854-3

Jaroszewicz S, Simovici DA (2004) Interestingness of frequent itemsets using bayesian networks as background knowledge. In: Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'04, ACM, pp 178–186, DOI 10.1145/1014052.1014074

Jiang M, Faloutsos C, Han J (2016) CatchTartan: Representing and summarizing dynamic multicontextual behaviors. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'16, ACM, pp 945–954, DOI 10.1145/2939672.2939749

Jonyer I, Holder LB, Cook DJ (2004) Mdl-based context-free graph grammar induction and applications. International Journal on Artificial Intelligence Tools 13(1):65–79, DOI 10.1142/S0218213004001429

Kameya Y (2011) Time series discretization via MDL-based histogram density estimation. In: Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI'11, IEEE Computer Society, pp 732–739, DOI 10.1109/ICTAI.2011.115

Kang U, Faloutsos C (2011) Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In: Proceedings of the 11th IEEE Inter-

national Conference on Data Mining, ICDM'11, IEEE Computer Society, pp 300–309, DOI 10.1109/ICDM.2011.26

Ketkar NS, Holder LB, Cook DJ (2005) Subdue: compression-based frequent pattern discovery in graph data. In: Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations, OSDM'05, ACM, pp 71–76, DOI 10.1145/1133905.1133915

Khan KU (2015) Set-based approach for lossless graph summarization using locality sensitive hashing. In: Proceedings of the 31st IEEE International Conference on Data Engineering Workshops, ICDEW'15, IEEE Computer Society, pp 255–259, DOI 10.1109/ICDEW.2015.7129586

Khan KU, Nawaz W, Lee YK (2014) Set-based unified approach for attributed graph summarization. In: Proceedings of the 4th IEEE International Conference on Big Data and Cloud Computing, BDCloud'14, IEEE Computer Society, pp 378–385, DOI 10.1109/BDCloud.2014.108

Khan KU, Nawaz W, Lee YK (2015a) Lossless graph summarization using dense subgraphs discovery. In: Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication, IMCOM'15, ACM, pp 1–7, DOI 10.1145/2701126.2701157

Khan KU, Nawaz W, Lee YK (2015b) Set-based approximate approach for lossless graph summarization. Computing 97(12):1185–1207, DOI 10.1007/s00607-015-0454-9

Kiernan J, Terzi E (2008) Constructing comprehensive summaries of large event sequences. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'08, ACM, pp 417–425, DOI 10.1145/1401890.1401943

Kiernan J, Terzi E (2009a) Constructing comprehensive summaries of large event sequences. ACM Transactions on Knowledge Discovery from Data 3(4):21:1–21:31, DOI 10.1145/1631162.1631169

Kiernan J, Terzi E (2009b) EventSummarizer: A tool for summarizing large event sequences. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT'09, ACM, pp 1136–1139, DOI 10.1145/1516360.1516497

Koivisto M, Perola M, Varilo T, Hennah W, Ekelund J, Lukk M, Peltonen L, Ukkonen E, Mannila H (2002) An MDL method for finding haplotype blocks and for estimating the strength of haplotype block boundaries. In: Proceedings of the 2003 Pacific Symposium on Biocomputing, PSB'03, World Scientific, pp 502–513, DOI 10.1142/9789812776303_0047

Kontkanen P, Myllymäki P (2007) MDL histogram density estimation. In: Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS'07, pp 219–226

Kontonasios KN, De Bie T (2012) Formalizing complex prior information to quantify subjective interestingness of frequent pattern sets. In: Proceedings of the 11th International Symposium on Advances in Intelligent Data Analysis, IDA'12, Springer, pp 161–171

Kontonasios KN, Vreeken J, De Bie T (2013) Maximum entropy models for iteratively identifying subjectively interesting structure in real-valued data.

In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'13, Springer, pp 256–271

Koopman A, Siebes A (2008) Discovering relational item sets efficiently. In: Proceedings of the 2008 SIAM International Conference on Data Mining, SDM'08, SIAM, pp 108–119, DOI 10.1137/1.9781611972788.10

Koopman A, Siebes A (2009) Characteristic relational patterns. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'09, ACM, pp 437–446, DOI 10.1145/1557019.1557071

Koutra D, Kang U, Vreeken J, Faloutsos C (2014) VOG: Summarizing and understanding large graphs. In: Proceedings of the 2014 SIAM International Conference on Data Mining, SDM'14, SIAM, pp 91–99, DOI 10.1137/1.9781611973440.11

Koutra D, Kang U, Vreeken J, Faloutsos C (2015) Summarizing and understanding large graphs. Statistical Analysis and Data Mining 8(3):183–202

Lakshmanan LVS, Ng RT, Wang CX, Zhou X, Johnson TJ (2002) The generalized MDL approach for summarization. In: Proceedings of the 28th international conference on Very Large Data Bases, VLDB'02, VLDB Endowment, pp 766–777

Lam HT, Mörchen F, Fradkin D, Calders T (2012) Mining compressing sequential patterns. In: Proceedings of the 2012 SIAM International Conference on Data Mining, SDM'12, SIAM, pp 319–330, DOI 10.1137/1.9781611972825.28

Lam HT, Calders T, Yang J, Mörchen F, Fradkin D (2013) Zips: Mining compressing sequential patterns in streams. In: Proceedings of the Workshop on Interactive Data Exploration and Analytics, IDEA @KDD'13, ACM, pp 54–62, DOI 10.1145/2501511.2501520

Lam HT, Kiseleva J, Pechenizkiy M, Calders T (2014a) Decomposing a sequence into independent subsequences using compression algorithms. In: Proceedings of the Workshop on Interactive Data Exploration and Analytic, IDEA @KDD'14, pp 67–75

Lam HT, Mörchen F, Fradkin D, Calders T (2014b) Mining compressing sequential patterns. Statistical Analysis and Data Mining 7(1):34–52, DOI 10.1002/sam.11192

Lee K, Jo H, Ko J, Lim S, Shin K (2020) SSumM: Sparse summarization of massive graphs. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'20, ACM, pp 144–154, DOI 10.1145/3394486.3403057

LeFevre K, Terzi E (2010) GraSS: Graph structure summarization. In: Proceedings of the 2010 SIAM International Conference on Data Mining, SDM'10, SIAM, pp 454–465, DOI 10.1137/1.9781611972801.40

Lim Y, Kang U, Faloutsos C (2014) SlashBurn: Graph compression and mining beyond caveman communities. IEEE Transactions on Knowledge and Data Engineering 26(12):3077–3089, DOI 10.1109/TKDE.2014.2320716

Liu Y, Shah N, Koutra D (2015) An empirical comparison of the summarization power of graph clustering methods. arXiv:1511.06820

Liu Y, Safavi T, Shah N (2016) Reducing million-node graphs to a few structural patterns: A unified approach. In: Proceedings of the 12th International Workshop on Mining and Learning with Graphs, MLG @KDD'16, p 8

Liu Y, Safavi T, Dighe A, Koutra D (2018a) Graph summarization methods and applications: A survey. ACM Computing Surveys 51(3):62:1–62:34, DOI 10.1145/3186727

Liu Y, Safavi T, Shah N, Koutra D (2018b) Reducing large graphs to small supergraphs: a unified approach. Social Network Analysis and Mining 8(1):17, DOI 10.1007/s13278-018-0491-4

Lucchese C, Orlando S, Perego R (2010a) A generative pattern model for mining binary datasets. In: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC'10, ACM, pp 1109–1110, DOI 10.1145/1774088.1774320

Lucchese C, Orlando S, Perego R (2010b) Mining top-k patterns from binary datasets in presence of noise. In: Proceedings of the 2007 SIAM International Conference on Data Mining, SDM'07, SIAM, pp 165–176, DOI 10.1137/1.9781611972801.15

Lucchese C, Orlando S, Perego R (2014) A unifying framework for mining approximate top-k binary patterns. IEEE Transactions on Knowledge and Data Engineering 26(12):2900–2913, DOI 10.1109/TKDE.2013.181

Makhalova T (2021) Contributions to pattern set mining : from complex datasets to significant and useful pattern sets. Phd thesis, Université de Lorraine, URL `https://hal.univ-lorraine.fr/tel-03342124`

Makhalova T, Trnecka M (2019) From-below boolean matrix factorization algorithm based on MDL. arXiv:1901.09567

Makhalova T, Trnecka M (2021) From-below boolean matrix factorization algorithm based on MDL. Advances in Data Analysis and Classification 15(1):37–56, DOI 10.1007/s11634-019-00383-6

Makhalova T, Kuznetsov SO, Napoli A (2018a) A first study on what MDL can do for FCA. In: Proceedings of the Fifteen International Conference on Concept Lattices and Their Applications, CLA'18, pp 25–36

Makhalova T, Kuznetsov SO, Napoli A (2018b) MDL for FCA: Is there a place for background knowledge? In: Proceedings of the 6th International Workshop "What can FCA do for Artificial Intelligence?" @ IJCAI/ECAI'18, CEUR Workshop Proceedings, vol 2149, pp 45–56, URL `http://ceur-ws.org/Vol-2149/paper5.pdf`

Makhalova T, Kuznetsov SO, Napoli A (2019a) Numerical pattern mining through compression. In: Proceedings of the Data Compression Conference, DCC'19, pp 112–121, DOI 10.1109/DCC.2019.00019

Makhalova T, Kuznetsov SO, Napoli A (2019b) On coupling FCA and MDL in pattern mining. In: Proceedings of the international conference on Formal Concept Analysis, FCA'19, Springer, pp 332–340, DOI 10.1007/978-3-030-21462-3_23

Makhalova T, Kuznetsov SO, Napoli A (2020) Mint: MDL-based approach for mining INTeresting numerical pattern sets. arXiv:2011.14843

Makhalova T, Kuznetsov SO, Napoli A (2021) Likely-occurring itemsets for pattern mining. In: Proceedings of the 6th International Workshop "What can FCA do for Artificial Intelligence?" @ IJCAI'21, CEUR Workshop Proceedings, vol 2972, pp 39–50, URL http://ceur-ws.org/Vol-2972/paper4.pdf

Makhalova T, Kuznetsov SO, Napoli A (2022) Mint: MDL-based approach for mining INTeresting numerical pattern sets. Data Mining and Knowledge Discovery 36(1):108–145, DOI 10.1007/s10618-021-00799-9

Mampaey M (2010) Mining non-redundant information-theoretic dependencies between itemsets. In: Proceedings of the 12th International Conference on Data Warehousing and Knowledge Discovery, DaWaK'10, Springer, pp 130–141, DOI 10.1007/978-3-642-15105-7_11

Mampaey M, Vreeken J (2010) Summarising data by clustering items. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'10, pp 321–336, DOI 10.1007/978-3-642-15883-4_21

Mampaey M, Vreeken J (2013) Summarizing categorical data by clustering attributes. Data Mining and Knowledge Discovery 26(1):130–173, DOI 10.1007/s10618-011-0246-6

Mampaey M, Tatti N, Vreeken J (2011) Tell me what i need to know: succinctly summarizing data with itemsets. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'11, ACM, pp 573–581, DOI 10.1145/2020408.2020499

Mampaey M, Vreeken J, Tatti N (2012) Summarizing data succinctly with the most informative itemsets. ACM Transactions on Knowledge Discovery from Data 6(4):16:1–16:42, DOI 10.1145/2382577.2382580

Mannila H, Toivonen H, Verkamo AI (1994) Efficient algorithms for discovering association rules. In: Proceedings of the KDD Workshop, Association for the Advancement of Artificial Intelligence, pp 181–192

Mannila H, Koivisto M, Perola M, Varilo T, Hennah W, Ekelund J, Lukk M, Peltonen L, Ukkonen E (2003) Minimum description length block finder, a method to identify haplotype blocks and to compare the strength of block boundaries. The American Journal of Human Genetics 73(1):86–94, DOI 10.1086/376438

Markham TS, Evans S, Impson J, Steinbrecher E (2009) Implementation of an incremental MDL-based two part compression algorithm for model inference. In: Proceedings of the 2009 Data Compression Conference, DCC'09, pp 322–331, DOI 10.1109/DCC.2009.66

Matsubara Y, Sakurai Y, Faloutsos C (2014) AutoPlait: automatic mining of co-evolving time sequences. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD'14, ACM, pp 193–204, DOI 10.1145/2588555.2588556

Miettinen P, Vreeken J (2011) Model order selection for boolean matrix factorization. In: Proceedings of the 17th ACM SIGKDD international conference

on Knowledge discovery and data mining, KDD'11, ACM, pp 51–59, DOI 10.1145/2020408.2020424

Miettinen P, Vreeken J (2014) MDL4BMF: Minimum description length for boolean matrix factorization. ACM Transactions on Knowledge Discovery from Data 8(4):18:1–18:31, DOI 10.1145/2601437

Mitra S, Sastry PS (2019) Summarizing event sequences with serial episodes: A statistical model and an application. arXiv:1904.00516

Navlakha S, Rastogi R, Shrivastava N (2008) Graph summarization with bounded error. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD'08, ACM, pp 419–432, DOI 10.1145/1376616.1376661

Nguyen HV, Müller E, Vreeken J, Böhm K (2014) Unsupervised interaction-preserving discretization of multivariate data. Data Mining and Knowledge Discovery 28(5):1366–1397, DOI 10.1007/s10618-014-0350-5

Otaki K, Yamamoto A (2015) Edit operations on lattices for MDL-based pattern summarization. In: Proceedings of the International Workshop on Formal Concept Analysis and Applications @ICFCA'15

Papadimitriou S, Gionis A, Tsaparas P, Väisänen RA, Mannila H, Faloutsos C (2005) Parameter-free spatial data mining using MDL. In: Proceedings of the 5th IEEE International Conference on Data Mining, ICDM'05, IEEE Computer Society, pp 346–353, DOI 10.1109/ICDM.2005.117

Papadimitriou S, Sun J, Faloutsos C, Yu PS (2008) Hierarchical, parameter-free community discovery. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'08, Springer, pp 170–187, DOI 10.1007/978-3-540-87481-2_12

Phan NH, Ienco D, Poncelet P, Teisseire M (2013) Mining representative movement patterns through compression. In: Advances in Knowledge Discovery and Data Mining, Springer, pp 314–326, DOI 10.1007/978-3-642-37453-1_26

Plant C, Biedermann S, Böhm C (2020) Data compression as a comprehensive framework for graph drawing and representation learning. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'20, ACM, pp 1212–1222, DOI 10.1145/3394486.3403174

Prakash BA, Vreeken J, Faloutsos C (2014) Efficiently spotting the starting points of an epidemic in a large graph. Knowledge and Information Systems 38(1):35–59, DOI 10.1007/s10115-013-0671-5

Proença HM, van Leeuwen M (2020a) Interpretable multiclass classification by MDL-based rule lists. Information Sciences 512:1372–1393, DOI 10.1016/j.ins.2019.10.050

Proença HM, van Leeuwen M (2020b) Interpretable multiclass classification by MDL-based rule lists. arXiv:1905.00328

Proença HM, Grünwald PD, Bäck T, van Leeuwen M (2020) Discovering outstanding subgroup lists for numeric targets using MDL. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'20

Proença HM, Bäck T, van Leeuwen M (2021a) Robust subgroup discovery. arXiv:2103.13686

Proença HM, Grünwald PD, Bäck T, van Leeuwen M (2021b) Discovering outstanding subgroup lists for numeric targets using MDL. arXiv:2006.09186

Puolamäki K, Oikarinen E, Kang B, Lijffijt J, De Bie T (2020) Interactive visual data exploration with subjective feedback: an information-theoretic approach. Data Mining and Knowledge Discovery 34(1):21–49, DOI 10.1007/s10618-019-00655-x

Rakthanmanon T, Keogh EJ, Lonardi S, Evans S (2011) Time series epenthesis: Clustering time series streams requires ignoring some data. In: Proceedings of the 11th IEEE International Conference on Data Mining, ICDM'11, IEEE Computer Society, pp 547–556, DOI 10.1109/ICDM.2011.146

Rakthanmanon T, Keogh EJ, Lonardi S, Evans S (2012) MDL-based time series clustering. Knowledge and Information Systems 33(2):371–399, DOI 10.1007/s10115-012-0508-7

Rashidi P, Cook DJ (2013) COM: A method for mining and monitoring human activity patterns in home-based health monitoring systems. ACM Transactions on Intelligent Systems and Technology 4(4):64:1–64:20, DOI 10.1145/2508037.2508045

Rissanen J (1978) Modeling by shortest data description. Automatica 14(5):465–471, DOI 10.1016/0005-1098(78)90005-5

Rojas A, Calatayud J, Kowalewski M, Neuman M, Rosvall M (2021) A multiscale view of the phanerozoic fossil record reveals the three major biotic transitions. Communications Biology 4(1):1–8, DOI 10.1038/s42003-021-01805-y

Rosvall M, Bergstrom CT (2007) An information-theoretic framework for resolving community structure in complex networks. Proceedings of the National Academy of Sciences 104(18):7327–7331, DOI 10.1073/pnas.0611034104

Rosvall M, Bergstrom CT (2008) Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences 105(4):1118–1123, DOI 10.1073/pnas.0706851105

Rosvall M, Bergstrom CT (2010) Mapping change in large networks. PLoS ONE 5(1):1–7, DOI 10.1371/journal.pone.0008694

Rosvall M, Bergstrom CT (2011) Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. PLOS ONE 6(4):e18209, DOI 10.1371/journal.pone.0018209

Rosvall M, Axelsson D, Bergstrom CT (2009) The map equation. The European Physical Journal Special Topics 178(1):13–23, DOI 10.1140/epjst/e2010-01179-1

Sampson O, Berthold MR (2014) Widened KRIMP: Better performance through diverse parallelism. In: Proceedings of the 13th International Symposium on Advances in Intelligent Data Analysis, IDA'14, Springer, pp 276–285, DOI 10.1007/978-3-319-12571-8_24

Saran D, Vreeken J (2019) Summarizing dynamic graphs using MDL. Tech. rep., Saarland University

Shah N, Koutra D, Zou T, Gallagher B, Faloutsos C (2015) TimeCrunch: Interpretable dynamic graph summarization. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'15, ACM, pp 1055–1064, DOI 10.1145/2783258.2783321

Shah N, Koutra D, Jin L, Zou T, Gallagher B, Faloutsos C (2017) On summarizing large-scale dynamic graphs. IEEE Data Engineering Bulletin 40(3):75–88

Shannon CE (1948) A mathematical theory of communication. Bell System Technical Journal 27(3):379–423, DOI 10.1002/j.1538-7305.1948.tb01338.x

Shokoohi-Yekta M, Chen Y, Campana B, Hu B, Zakaria J, Keogh E (2015) Discovery of meaningful rules in time series. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'15, ACM, pp 1085–1094, DOI 10.1145/2783258.2783306

Siebes A (2012) Queries for data analysis. In: Proceedings of the 11th International Symposium on Advances in Intelligent Data Analysis, IDA'12, Springer, pp 7–22

Siebes A (2014) MDL in pattern mining: A brief introduction to krimp. In: Proceedings of the international conference on Formal Concept Analysis, FCA'14, Springer, pp 37–43, DOI 10.1007/978-3-319-07248-7_3

Siebes A, Kersten R (2011) A structure function for transaction data. In: Proceedings of the 2011 SIAM International Conference on Data Mining, SDM'11, SIAM, pp 558–569, DOI 10.1137/1.9781611972818.48

Siebes A, Kersten R (2012) Smoothing categorical data. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'12, Springer, pp 42–57, DOI 10.1007/978-3-642-33460-3_8

Siebes A, Vreeken J, van Leeuwen M (2006) Item sets that compress. In: Proceedings of the 2006 SIAM International Conference on Data Mining, SDM'06, SIAM

Smets K, Vreeken J (2011) The odd one out: Identifying and characterising anomalies. In: Proceedings of the 2011 SIAM International Conference on Data Mining, SDM'11, SIAM, pp 804–815, DOI 10.1137/1.9781611972818.69

Smets K, Vreeken J (2012) Slim: Directly mining descriptive patterns. In: Proceedings of the 2012 SIAM International Conference on Data Mining, SDM'12, SIAM, pp 236–247

Soulet A, Raïssi C, Plantevit M, Crémilleux B (2011) Mining dominant patterns in the sky. In: Proceedings of the 11th IEEE International Conference on Data Mining, ICDM'11, IEEE Computer Society, pp 655–664, DOI 10.1109/ICDM.2011.100

Stone JV (2013) Information Theory: A Tutorial Introduction. Sebtel Press

Sun J, Faloutsos C, Papadimitriou S, Yu PS (2007) GraphScope: parameter-free mining of large time-evolving graphs. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'07, ACM, pp 687–696, DOI 10.1145/1281192.1281266

Tanaka Y, Uehara K (2003) Discover motifs in multi-dimensional time-series using the principal component analysis and the MDL principle. In: Proceedings of the 3rd international conference on Machine learning and data mining in pattern recognition, MLDM'03, Springer, pp 252–265

Tanaka Y, Iwamoto K, Uehara K (2005) Discovery of time-series motif from multi-dimensional data based on MDL principle. Machine Learning 58(2):269–300, DOI 10.1007/s10994-005-5829-2

Tatti N (2010) Probably the best itemsets. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'10, ACM, pp 293–302, DOI 10.1145/1835804.1835843

Tatti N, Heikinheimo H (2008) Decomposable families of itemsets. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'08, pp 472–487, DOI 10.1007/978-3-540-87481-2_31

Tatti N, Vreeken J (2008) Finding good itemsets by packing data. In: Proceedings of the 8th IEEE International Conference on Data Mining, ICDM'08, IEEE Computer Society, pp 588–597, DOI 10.1109/ICDM.2008.39

Tatti N, Vreeken J (2012a) Discovering descriptive tile trees. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'12, Springer, pp 9–24, DOI 10.1007/978-3-642-33460-3_6

Tatti N, Vreeken J (2012b) The long and the short of it: Summarising event sequences with serial episodes. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'12, ACM, pp 462–470

van Leeuwen M (2010) Patterns that matter. Phd thesis, Universiteit Utrecht

van Leeuwen M, Galbrun E (2015) Association discovery in two-view data. IEEE Transactions on Knowledge and Data Engineering 27(12):3190–3202, DOI 10.1109/TKDE.2015.2453159

van Leeuwen M, Siebes A (2008) StreamKrimp: Detecting change in data streams. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'08, Springer, pp 672–687, DOI 10.1007/978-3-540-87479-9_62

van Leeuwen M, Vreeken J (2014) Mining and using sets of patterns through compression. In: Frequent Pattern Mining, Springer, pp 165–198, DOI 10.1007/978-3-319-07821-2_8

van Leeuwen M, Vreeken J, Siebes A (2006) Compression picks item sets that matter. In: Proceedings of the European Conference on Knowledge Discovery in Databases, PKDD'06, Springer, pp 585–592, DOI 10.1007/11871637_59

van Leeuwen M, Bonchi F, Sigurbjörnsson B, Siebes A (2009a) Compressing tags to find interesting media groups. In: Proceedings of the 18th ACM conference on Information and knowledge management, CIKM'09, ACM, pp 1147–1156, DOI 10.1145/1645953.1646099

van Leeuwen M, Vreeken J, Siebes A (2009b) Identifying the components. Data Mining and Knowledge Discovery 19(2):176–193, DOI 10.1007/s10618-009-0137-2

van Leeuwen M, De Bie T, Spyropoulou E, Mesnage C (2016) Subjective interestingness of subgraph patterns. Machine Learning 105(1):41–75, DOI 10.1007/s10994-015-5539-3

Vanetik N, Litvak M (2017) Query-based summarization using MDL principle. In: Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres @ACL'17, pp 22–31

Vanetik N, Litvak M (2018) DRIM: MDL-based approach for fast diverse summarization. In: Proceedings of the 2018 IEEE/WIC/ACM International Conference on Web Intelligence, WI'18, pp 660–663, DOI 10.1109/WI.2018.00-17

Vespier U, Knobbe A, Nijssen S, Vanschoren J (2012) MDL-based analysis of time series at multiple time-scales. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD'12, Springer, pp 371–386, DOI 10.1007/978-3-642-33486-3_24

Viamontes Esquivel A, Rosvall M (2011) Compression of flow can reveal overlapping-module organization in networks. Physical Review X 1(2):021025, DOI 10.1103/PhysRevX.1.021025

Vreeken J (2009) Making pattern mining useful. Phd thesis, Universiteit Utrecht

Vreeken J, Siebes A (2008) Filling in the blanks – krimp minimisation for missing data. In: Proceedings of the 8th IEEE International Conference on Data Mining, ICDM'08, IEEE Computer Society, pp 1067–1072, DOI 10.1109/ICDM.2008.40

Vreeken J, van Leeuwen M, Siebes A (2007a) Characterising the difference. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'07, ACM, pp 765–774, DOI 10.1145/1281192.1281274

Vreeken J, van Leeuwen M, Siebes A (2007b) Preserving privacy through data generation. In: Proceedings of the 7th IEEE International Conference on Data Mining, ICDM'07, IEEE Computer Society, pp 685–690, DOI 10.1109/ICDM.2007.25

Vreeken J, van Leeuwen M, Siebes A (2011) Krimp: Mining itemsets that compress. Data Mining and Knowledge Discovery 23(1):169–214

Wang P, Wang H, Liu M, Wang W (2010) An algorithmic approach to event summarization. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD'10, ACM, pp 183–194, DOI 10.1145/1807167.1807189

Webb GI (2007) Discovering significant patterns. Machine Learning 68(1):1–33, DOI 10.1007/s10994-007-5006-x

Webb GI, Vreeken J (2013) Efficient discovery of the most interesting associations. ACM Transactions on Knowledge Discovery from Data 8(3):15:1–15:31, DOI 10.1145/2601433

Wiegand B, Klakow D, Vreeken J (2021) Mining easily understandable models from complex event logs. In: Proceedings of the 2021 SIAM International Conference on Data Mining, SDM'21, SIAM, pp 244–252, DOI 10.1137/1.9781611976700.28

Wiegand B, Klakow D, Vreeken J (2022) Mining interpretable data-to-sequence generators. In: Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI'22, Association for the Advancement of Artificial Intelligence

Witteveen J, Duivesteijn W, Knobbe A, Grünwald PD (2014) RealKrimp – finding hyperintervals that compress with MDL for real-valued data. In: Proceedings of the 13th International Symposium on Advances in Intelligent Data Analysis, IDA'14, Springer, pp 368–379, DOI 10.1007/978-3-319-12571-8_32

Wu D, Gundimeda S, Mou S, Quinn CJ (2020) Modeling piece-wise stationary time series. In: Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'20, IEEE Computer Society, pp 3817–3821, DOI 10.1109/ICASSP40776.2020.9053470

Yan X, Cheng H, Han J, Xin D (2005) Summarizing itemset patterns: a profile-based approach. In: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'05, ACM, pp 314–323, DOI 10.1145/1081870.1081907

Yan Y, Cao L, Madden S, Rundensteiner EA (2018) SWIFT: Mining representative patterns from large event streams. Proc VLDB Endow 12(3):265–277, DOI 10.14778/3291264.3291271

Yang L, Baratchi M, van Leeuwen M (2020) Unsupervised discretization by two-dimensional MDL-based histogram. arXiv:2006.01893

Youngblood GM, Heierman EO, Cook DJ, Holder LB (2005) Automated HPOMDP construction through data-mining techniques in the intelligent environment domain. In: Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, FLAIRS'05

Yurov M, Ignatov DI (2017) Turning krimp into a triclustering technique on sets of attribute-condition pairs that compress. In: Proceedings of the International Joint Conference on Rough Sets, IJCRS'17, Springer, pp 558–569, DOI 10.1007/978-3-319-60840-2_40

Zhao P, Zhao Q, Zhang C, Su G, Zhang Q, Rao W (2019) CLEAN: Frequent pattern-based trajectory spatial-temporal compression on road networks. In: Proceedings of the 20th IEEE International Conference on Mobile Data Management, MDM'19, IEEE Computer Society, pp 605–610, DOI 10.1109/MDM.2019.00127