Mining Local Staircase Patterns in Noisy Data

Thanh Le Van*, Ana Carolina Fierro†, Tias Guns*, Matthijs van Leeuwen*, Siegfried Nijssen*,
Luc De Raedt*, Kathleen Marchal‡
*Department of Computer Science, KU Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be
†Department of Microbial and Molecular Systems, KU Leuven, Belgium
Carolina.Fierro@biw.kuleuven.be
‡Department of Plant Systems Biology, VIB, Belgium
Department of Plant Biotechnology and Bioinformatics, Ghent University, Belgium
kamar@psb.ugent.be

***Abstract*—Most traditional biclustering algorithms identify biclusters with no or little overlap. In this paper, we introduce the problem of identifying staircases of biclusters. Such staircases may be indicative for causal relationships between columns and can not easily be identified by existing biclustering algorithms. Our formalization relies on a scoring function based on the Minimum Description Length principle. Furthermore, we propose a first algorithm for identifying staircase biclusters, based on a combination of local search and constraint programming. Experiments show that the approach is promising.**

***Index Terms*—Staircase patterns; pattern sets; constraint programming; MDL; biclustering.**

I. Introduction

A bicluster traditionally consists of a subset of attributes together with a subset of examples in which the data shows a regularity of interest [4]. For example, in the case of *constant row biclustering*, a bicluster identifies a submatrix in which each row approximately has a constant value. Such biclusters can be seen as rectangular subsets of a given data matrix, and are indicative of associations between attributes and examples. The task of biclustering then usually consists of identifying a set of non-overlapping biclusters that together characterize a data set well.

There are limitations to the applicability of this traditional biclustering model; the following toy example, which is illustrative for a larger number of applications, highlights this. Let us assume that for a large number of broken electronical devices (considered to be the rows or examples in our data), it is recorded which components are no longer working (components hence correspond to columns in the data). Within this data, one can expect that for subsets of devices and for subsets of components, cascades of failures are observed: for instance, if the power supply of a device breaks down, it is likely that many other components dependent on the power supply will fail as well. The result is a cascade of failures.

Characteristic of such failure cascades is that:

- they only apply to a subset of examples and attributes;
- the subset of data within the cascade has a nested structure: we can order the devices such that the failures of a device are a subset of the failures of the previous device. Visually, hence, the cascades look like *staircases* instead of rectangles.

Similar tasks can be identified in paleontology (where staircases in presence/absence data of species may reflect a predators-prey food chain), distribution networks (where staircases reflect distribution channels for certain types of items), social networks (where staircases reflect cascades of news distribution for certain types of news), the analysis of gene expression data (where a staircase can present evidence of the participation of genes in a perturbed biological pathway), and the analysis of educational data (where staircases represent dependencies between courses for certain groups of students). In some applications, it may be of interest to study data which is not binary: in bioinformatics, genes may be over-expressed or under-expressed; in distribution networks, it could be important to distinguish between variations of a single product being distributed.

To the best of our knowledge, existing biclustering systems can not discover such staircases; the main reason is that staircases can be seen as sets of related, highly overlapping biclusters; most biclustering algorithms assume no or few overlap between biclusters [12]. An example is provided in Figure 1, where a staircase is given consisting of 3 highly overlapping biclusters. The problem is also different from the discovery of nested segments [13], as we assume that the staircases are only *local* phenomena present in subsets of instances, and we do not limit ourselves to binary data.

Summarizing, we are interested in discovering biclusters that satisfy the following requirements:

- they accurately describe a large part of the data matrix;
- each next bicluster in a staircase includes more columns, and less rows, compared to the previous bicluster;
- each row in a bicluster approximately has a constant value.

The contribution of this paper is threefold. First, we introduce the problem of finding staircases in non-binary data. Second, we formalize the quality of a staircase by means of the *Minimum Description Length (MDL) principle*. Finally, we use constraint programming to search for staircases. Both the MDL score and the constraint programming model take into account the high-level requirements listed above.

The main computational challenge is that directly finding staircases that optimize the MDL score is hard. In this paper,

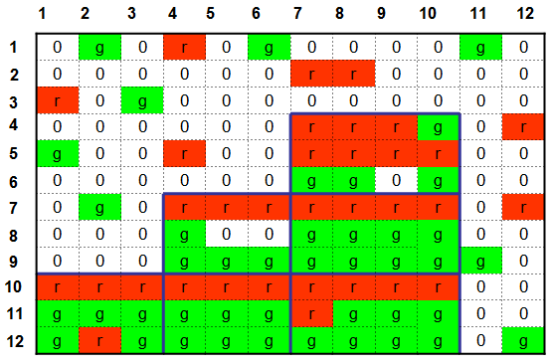


Fig. 1: An example of a staircase consisting of three fault-tolerant constant-row biclusters, the first bicluster ranges over columns 7–10 and rows 4–12, the second over columns 4–10 and rows 7–12 the, third over columns 1–10 and rows 10–12.

we study a two-phased heuristic approach to address this problem. This approach separates the generation of candidate staircases from the evaluation of the staircases. During the generation of candidates we take into account the same high-level requirements that the MDL score is based on, but we do not evaluate the MDL score itself.

To model these high level requirements, we propose an approach using *constraint programming* (CP) [7], [10]. Constraint programming systems find solutions to problems formalized as *constraint satisfaction problems*. In our case, the constraints are the high-level requirements and the solutions form candidate staircases.

An important reason for proposing to use CP and the MDL principle is that both approaches are general and principled approaches to problem solving. Consequently, we believe that the methodology presented in this paper is more easily extended to a wide range of related settings than an adhoc solution would be.

The rest of the paper is organized as follows: Section II presents the theoretical problem formulation, Section III presents the first phase of our method; mining staircases using constraint programming, while Section IV presents the second phase: how to select the best staircase using an MDL-based score. Finally, Section V presents the experiments and related work and conclusions are discussed in the last two sections.

II. Staircase pattern sets

We will first introduce the constraints that we use to identify candidate staircases. We do this by first defining a fault-tolerant bicluster in terms of constraints, and then propose a formalization of the problem of finding a set of biclusters that together form a *staircase* pattern set.

A. Constant-row biclusters

Let $\sigma = \{0, a, b, c, \ldots\}$ be a finite set of symbols where 0 represents a special *neutral* value and $a, b, c, \ldots$ are arbitrary symbols. Let dataset $\mathcal{D}$ be an $m \times n$ matrix with m rows and n columns. Let $M = \{1, \ldots, m\}$ and $N = \{1, \ldots, n\}$. Each cell in the matrix contains a symbol from σ: $\forall i, j : \mathcal{D}_{ij} \in \sigma$. With σ^+ we will denote the set of symbols *without* 0: $\sigma^+ = \sigma \backslash \{0\}$.

A *bicluster* B is a tuple (R, C) where $R \subseteq M$ is a subset of the rows and $C \subseteq N$ a subset of the columns. The name biclusters stems from the fact that it can be seen as a cluster on both the rows and the columns.

Definition 2.1 (Constant-row bicluster): Given a matrix $\mathcal{D}$ over symbols σ, a *constant-row bicluster* is a tuple $B = (R, C)$ where for each row in R, on that row all columns in C have the same (non-zero) symbol:

$$\forall r \in R : \exists s \in \sigma^+ \text{ s.t. } \forall c \in C, \mathcal{D}_{r,c} = s$$

In the rest of this paper, when we write bicluster we implicitly assume a constant-row bicluster.

In case there are only two symbols $\sigma = \{0, 1\}$, the above definition of a constant-row bicluster matches the definitions of a *tile* and an *itemset* [7]. Similar to itemsets, we can now define the *cover* relation between columns and rows:

Definition 2.2 (Cover of a set of columns): Given a set of columns C and a matrix $\mathcal{D}$ over symbols σ, the cover of C is the set of rows for which the constant row property over C holds:

$$\varphi_{\mathcal{D}}(C) = \{r \in M \mid \exists s \in \sigma^+ : \forall c \in C, \mathcal{D}_{r,c} = s\} \quad (1)$$

The above formulation requires that every row consists of exactly the same symbol. However, many datasets contain *noise*, that is, a fraction of the elements have a deviating symbol. We can cater for this by introducing the concept of fault-tolerance [2]. A fault-tolerant bicluster is a bicluster that allows a small amount of noise on the rows and columns. In the following, we use the *Iverson bracket* $[\cdot]$ to convert the truth value of an equation into 1/0, e.g. $[8 \geq 5] = [true] = 1, [5 \geq 8] = [false] = 0$.

Definition 2.3 (Fault-tolerant cover): Given a set of columns C and a matrix $\mathcal{D}$ over symbols σ, as well as a threshold ϵ_R, the fault-tolerant cover of C is the set of rows in which at least ϵ_R percent of the columns in C have the same symbol:

$$\varphi^{ft}_{\mathcal{D}}(C, \epsilon_R) = \{r \in M \mid \exists s \in \sigma^+ : \frac{1}{|C|} \sum_{c \in C} [\mathcal{D}_{r,c} = s] \geq \epsilon_R\} \quad (2)$$

Using the above definition, a fault-tolerant bicluster could have all its noise grouped into a few columns. A column with mostly noise should not be considered part of a bicluster, hence we constrain the noise in the columns as well:

Definition 2.4 (Fault-tolerant columns): Given a matrix $\mathcal{D}$ over symbols σ, a bicluster (R, C) as well as a threshold ϵ_C. A fault-tolerant column is a column in which at least ϵ_C percent of the symbols in R do not deviate from the constant-row symbol:

$$\Psi^{ft}_{\mathcal{D}}(R, C, \epsilon_C) = \{c \in C \mid \frac{1}{|R|} \sum_{r \in R} [\mathcal{D}_{r,c} = S_r] \geq \epsilon_C\} \quad (3)$$

where S_r is the constant-row symbol for r, i.e. the symbol that appears most frequently in row r for columns C:

$$\forall r \in R : S_r = \arg\max_{s \in \sigma^+} \sum_{c \in C} [\mathcal{D}_{r,c} = s] \quad (4)$$

We can now define a fault-tolerant bicluster as a bicluster whose rows are the fault-tolerant cover of the columns, and whose columns are fault-tolerant as well:

Definition 2.5 (Fault-tolerant bicluster): Given a matrix $\mathcal{D}$ over symbols σ and thresholds ϵ_R and ϵ_C. A fault tolerant bicluster $B = (R, C)$ satisfies the following properties:

$$R = \varphi^{ft}_{\mathcal{D}}(C, \epsilon_R), \quad (5)$$

$$C = \Psi^{ft}_{\mathcal{D}}(R, C, \epsilon_C) \quad (6)$$

A *frequent* fault-tolerant bicluster is a fault-tolerant bicluster for which $|R| \geq \theta$, that is, it contains more than θ rows where θ is a threshold.

B. Staircase of biclusters

As motivated in the introduction, we wish to find a staircase of fault-tolerant biclusters. Characteristic for a staircase is that it contains multiple *steps*. Each step removes part of the rows, but includes more columns. This intuition leads to the following definition.

Definition 2.6 (Fault-tolerant bicluster staircase): A staircase S is a set of fault-tolerant biclusters $S = \{B_1, B_2, \ldots, B_k\}$, with $B_i = (R_i, C_i)$ such that $R_1 \supset R_2 \supset \ldots \supset R_k$ and $C_1 \subset C_2 \subset \ldots \subset C_k$.

The *cover* of a staircase S consists of all elements in matrix $\mathcal{D}$ that fall within any of the biclusters. We identify an element of the matrix by its row and column index.

Definition 2.7 (Staircase cover): Given a staircase set S and a matrix $\mathcal{D}$ over symbols σ, the cover of S is given as:

$$cover_{\mathcal{D}}(S) = \{(r, c) \mid \exists (R_i, C_i) \in S : r \in R_i \wedge c \in C_i\} \quad (7)$$

For ease of presentation, we will denote with $\mathcal{D}^S$ that part of the matrix $\mathcal{D}$ that belongs to the cover of S, and with $\mathcal{D}^{\setminus S}$ its complement, i.e. all values in the matrix not belonging to the cover of S.

The problem that we consider in this paper is that of mining a staircase that fits the data well. Intuitively, this means that we want to find a staircase that 1) has a large cover, 2) has clearly discernible *steps*, 3) contains as little noise as possible. The requirement that we wish to find a staircase with large cover can be formalized as that we wish to maximize $cover_{\mathcal{D}}(S)$. One way to achieve discernible steps is to require that each step in the staircase covers a sufficiently large new area compared to the previous step, i.e. to require that $|cover_{\mathcal{D}}(\{B_1, \ldots, B_{i-1}, B_i\}) \setminus cover_{\mathcal{D}}(\{B_1, \ldots, B_{i-1}\})|$ is maximized for each bicluster B_i. Unfortunately, doing this would have an undesirable side effect. This is illustrated in Figure 2. Compared to $B_1 = (R_1, C_1)$, bicluster B_3 adds more to B_1 than B_2 adds to B_1; hence B_3 would be selected instead of B_2. To control the *size of the step*, we introduce a parameter θ and a constraint on the minimum number of rows that the next step has to cover. The threshold of the constraint is calculated by $|R_1| - \frac{\theta}{M} i$, where M is the total number of rows and i is incremented at each iteration of the algorithm. When no bicluster is found, i is incremented as well, thereby ensuring that steps larger than θ can also be found.

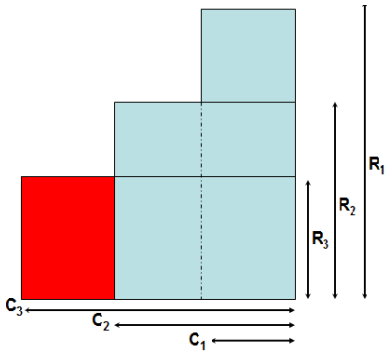


Fig. 2: A staircase pattern set consisting of three biclusters, i.e. $B_1 = (R_1, C_1), B_2 = (R_2, C_2), B_3 = (R_3, C_3)$. The area that B_3 adds to the staircase (indicated in red) can be calculated by $|R_3| * |C_3 \setminus C_2|$.

Overall we are looking for a staircase $S = \{B_1, B_2, \ldots, B_k\}$ that satisfies:

- each $B_i \in S$ is a fault-tolerant bicluster under thresholds ϵ_R and ϵ_C;
- for all $1 \leq i \leq k-1$: $R_{i+1} \subset R_i$ and $C_i \subset C_{i+1}$;
- $|R_1|$ is maximal;
- for each $i \in \{2, 3, \ldots, k\}$: $|cover_{\mathcal{D}}(\{B_1, \ldots, B_{i-1}, B_i\}) \setminus cover_{\mathcal{D}}(\{B_1, \ldots, B_{i-1}\})|$ is maximal, under the constraint that $|R_i| \geq |R_1| - \frac{\theta}{M} i$.

The advantage of this formalization, as we will see in the next section, is that computing staircases becomes more tractable. However, we should point out that the approach is dependent on the choice of parameter θ which controls the size of the *steps*; for an incorrect parameter size, we may miss parts of the staircase.

III. Finding Biclusters

Given our requirements on staircases, a greedy algorithm in which we first identify B_1 and subsequently identify $B_2, \ldots, B_k$ is the most logical choice.

Both for finding the first bicluster and the subsequent biclusters, we will employ constraint-based local search [10]. This search method has a non-deterministic component. We therefore run the search multiple times. From the staircases found, we choose one using an MDL-based score; this will be explained in Section IV.

We are building on the *constraint programming for itemset mining* framework proposed by De Raedt et al. [7]. This framework can more readily be applied on binary $\{0, 1\}$ data, hence we first transform the data to binary form.

A. Data transformation

Given a dataset $\mathcal{D}$ over σ, we split each row in $|\sigma^+|$ rows, one for each symbol. Such a row contains a value of 1 where the row had that symbol in the respective column, and 0 elsewhere:

$$\forall r \in \{1,\ldots,m\}:$$
$$\forall x \in \{1,\ldots,|\sigma^+|\}:$$
$$\forall c \in \{1,\ldots,n\}:$$
$$r' = (x + (r-1)|\sigma^+|),$$
$$\mathcal{D}'_{r',c} = \begin{cases} 1 & \text{if } \mathcal{D}_{r,c} = {\sigma^+}_x \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

By construction of the transformation, an itemset in this transformed binary database $\mathcal{D}'$ corresponds to a constant-row bicluster in the original data.

B. Fault-tolerant biclusters in CP

All biclusters in the staircase need to satisfy the fault-tolerance constraints. To express these constraints, we introduce a Boolean variable for every row $\mathcal{D}'$; $\forall t \in \mathcal{T} = \{1,\ldots,|\sigma^+| * m\} : T_t \in \{0,1\}$ and one for every column of $\forall i \in \mathcal{I} = \{1,\ldots,n\} : I_i \in \{0,1\}$. An assignment to the Boolean vectors T and I corresponds to one bicluster, by indicating the rows and columns that define the bicluster.

We can now express the cover of a set of columns (Definition 2.3) as a constraint on these variables as follows:

Theorem 3.1 (Fault-tolerant cover constraint): If the following constraint is satisfied:

$$\forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} (\mathcal{D}'_{t,i} - \epsilon_R) I_i \geq 0, \quad (9)$$

then the fault-tolerant cover set $\varphi^{ft}_{\mathcal{D}}(C, \epsilon_R)$ can be derived from T as follows:

$$\varphi^{ft}_{\mathcal{D}}(C, \epsilon_R) = \{r \in M \mid \exists x \in \{1,\ldots|\sigma^+|\} : T_{x+(r-1)|\sigma^+|} = 1\}.$$

The proof is given in the Appendix.

Fault-tolerant columns (Definition 2.4) can be expressed as a constraint as follows:

Theorem 3.2 (Fault-tolerant columns constraint): Assuming the constraint in Equation 9 is imposed, then the following two constraints enforce that $I_i = 1 \leftrightarrow i \in \Psi^{ft}_{\mathcal{D}}(R, C, \epsilon_C)$:

$$\forall r \in M : \sum_{x \in \{1,\ldots,|\sigma^+|\}} T_{(x+(r-1)*|\sigma^+|)} = 1 \quad (10)$$

$$\forall i \in \mathcal{I} : I_i = 1 \rightarrow \sum_{t \in \mathcal{T}} (\mathcal{D}'_{t,i} - \epsilon_C) T_t \geq 0 \quad (11)$$

A key insight here is that if Equation (10) holds then the $s = {\sigma^+}_x$ for which $T_{(x+(r-1)|\sigma^+|)} = 1$ is exactly $\arg\max_{s \in \sigma} \sum_{c \in C} [\mathcal{D}_{r,c} = s]$.

The combination of the constraints in Equations (9), (10) and (11) results in a constraint specification that can be used to find fault-tolerant biclusters.

a) Mining the first bicluster: In addition to the fault tolerance constraints, the first bicluster in the staircase should cover as many rows as possible. Observe that because $\sum_{x \in \{1,\ldots,|\sigma^+|\}} T_{(x+(r-1)*|\sigma^+|)} = 1$ we have that $|R| = \sum_{t \in \mathcal{T}} T_t$. Hence, the first bicluster has to satisfy the constraints in equations (9), (10) and (11) while maximizing

$$\sum_{t \in \mathcal{T}} T_t \quad (12)$$

b) Mining subsequent biclusters: For subsequent biclusters we need to employ a different optimization criterion. Remember that we want to maximize $|cover_{\mathcal{D}}(\{B_1,\ldots,B_{i-1}, B_i\}) \setminus cover_{\mathcal{D}}(\{B_1,\ldots,B_{i-1}\})|$. This corresponds to the red area in Figure 2. In the example of the figure, the size of the area is $|R_3| * |C_3 \setminus C_2|$. In general, the area that bicluster $B_k = (R_k, C_k)$ adds to a staircase of $k-1$ biclusters can be calculated by $|R_k| * |C_k \setminus C_{k-1}|$.

In the binarised data we want to maximize:

$$(\sum_{t \in \mathcal{T}} T_t)(\sum_{i \in \mathcal{I} \setminus C_{k-1}} I_i),$$

where C_{k-1} represents the columns selected in bicluster B_{k-1}, and T and I are vectors of boolean variables representing the bicluster B_k. Furthermore, several constraints are applied. First, the staircase constraints $C_{i+1} \supseteq C_i$ and $R_{i+1} \subseteq R_i$ need to be satisfied:

$$\forall i \in C_{k-1} : \quad I_i = 1$$
$$\forall r \in M \setminus R_{k-1} : \quad \forall x \in \{1,\ldots,|\sigma^+|\} : T_{x+(r-1)|\sigma^+|} = 0$$

Also, we need to satisfy the minimum row constraint that controls the size of the steps. A straight-forward translation of that constraint is $\sum_{t \in \mathcal{T}} T_t \geq |R_1| - \frac{\theta}{M} k$. However, in earlier work it was found that a *reified constraint* often works better [7], as it provides better pruning of the search space. We choose to reify the constraint here as well:

$$\forall i \in \mathcal{I} \setminus \cup_{j=1}^{k-1} I_j : \quad I_i = 1 \rightarrow \sum_{t \in \mathcal{T}} \mathcal{D}'_{t,i} T_t \geq |R_1| - \frac{\theta}{M} k;$$

Finally, we need to satisfy the fault tolerance constraints linking rows and columns. For the columns, we reuse constraints (10) and (11). For the rows, we adapt (9) so that it is only posted on the rows that have previously been covered:

$$\forall t \in \cup_{r \in R_{k-1}} \{1 + (r-1) * |\sigma^+|, \ldots, |\sigma^+| + (r-1) * |\sigma^+|\} :$$
$$T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} (\mathcal{D}'_{t,i} - \epsilon_R) I_i \geq 0$$

C. Large Neighbourhood Search

Mining a fault-tolerant bicluster is equivalent to finding an assignment for T and I that satisfies the required constraints and optimization criteria. The number of possible biclusters is exponential in the number of columns in the matrix, and searching for the best fault-tolerant bicluster is very time consuming. Hence, we use a form of local search to speed up the search.

Large Neighbourhood Search (LNS) is a hybridization of local search and constraint propagation in CP. Constraint propagation is a mechanism that ensures that the domains of variables are reduced if possible; for instance, in our biclustering formalization it ensures that row variables are fixed to the value 0 if the noise in rows is too large. While traditional local search methods choose a *neighbouring* solution by making some changes to a limited number of variables, LNS selects a subset of the problem (i.e. a random subset of variables) and performs complete search over these variable to find an optimal assignment. Two main questions involved with LNS include *a*) Which variables should be selected to search over; *b*) How to search on these variables. In our implementation, we use stochastic variable selection and exhaustive search approach as described in [8]. The stochastic variable selection randomly selects a subset of variables to search over. This has the risk of not finding the optimal solution, hence we run the algorithm multiple times. The next section explains how the best staircase is selected from this set of found staircases.

IV. Staircase selection by MDL

After producing multiple candidate staircases by running the Large Neighbourhood Search multiple times, the main question is how to pick the one that best fits the data. Since we are looking for a set of biclusters that together *describes* the data as accurately as possible, we employ the Minimum Description Length (MDL) principle to select the best staircase among the candidates. With this choice, we follow a recent trend in pattern mining [18].

The MDL principle states that the best model for given data is the one that compresses the data best. In the current context, the best staircase S^* is the one that minimizes the total compressed size:

$$L(\mathcal{D}, S) = L(S) + L(\mathcal{D} \mid S), \tag{13}$$

where $L(\mathcal{D} \mid S)$ is the size (in bits) of $\mathcal{D}$ encoded with S, and $L(S)$ is the size (in bits) of S.

Note that compression should be lossless, and that both the encoded size of the model and that of the data encoded with the model are taken into account, ensuring that model and data complexity are balanced. This way, we avoid 'overfitting' on the data by picking a staircase consisting of many biclusters. We previously already defined staircases, our models, and how they cover the data. What remains is that we need to determine how to encode a staircase S, and how to encode a matrix $\mathcal{D}$ with a staircase S.

A. Encoding a staircase

From the definition of a staircase, i.e. Definition 2.6, we know that the initial bicluster can be any bicluster, after which each subsequent bicluster is strongly constrained with regard to its predecessor. In the following, we exploit this observation to develop a succinct encoding for staircases.

1) The first bicluster: Given a staircase S, $B_1 = (R_1, C_1)$ is the initial bicluster and the one that we encode first. For this we use the n row and m column indexes assigned to the matrix, i.e. N and M. From these two sets, we encode those indexes that correspond to the rows and columns in B_1. From information theory, we have that we can encode an index $x \in [1, X]$ in $\log X$ bits. (All logarithms are to base 2.)

Apart from encoding the row and column indexes corresponding to the $|R_1|$ rows and the $|C_1|$ columns, we also need to encode the number of rows and the number of columns to ensure that lossless decoding is possible. This adds an extra $\log m + \log n$ bits and results in a total encoded size for bicluster B_1, denoted by $L(B_1)$, given by:

$$L(B_1) = (|R_1| + 1) \log m + (|C_1| + 1) \log n \tag{14}$$

2) Subsequent biclusters: For a succinct encoding of the remaining biclusters, we exploit the knowledge that each subsequent bicluster is *very similar* to the previous one. Instead of encoding the complete row and column sets, we only encode the *differences* with respect to the previous bicluster in the sequence. To be more precise, each $B_i = (R_i, C_i)$ with $1 < i \leq k$ is encoded relative to B_{i-1}. For each two consecutive biclusters, we only need to encode two types of modifications:

- *Remove rows*: $q = |R_{i-1} \setminus R_i|$ rows are removed. Each of these changes can be encoded in $\log(|R_{i-1}|)$ bits, since we only need to choose from the previous set of rows R_{i-1}.
- *Add columns*: $p = |C_i \setminus C_{i-1}|$ columns are added. Each of these changes can be encoded in $\log(N - |C_{i-1}|)$ bits, since we only need to choose from all columns not in C_{i-1}.

As before, we also need to encode how many changes are made, to ensure that lossless decoding is possible. Together, this gives the following encoded size for B_i:

$$\begin{aligned} L(B_i) \;&= (q+1)\log(|R_{i-1}|) \quad + \\ &\quad (p+1)\log(n - |C_{i-1}|), \quad 1 < i \leq k \end{aligned} \tag{15}$$

Finally, the total encoded size of staircase S is given by

$$L(S) = \sum_{i \in [1,k]} L(B_i). \tag{16}$$

B. Encoding the data

Since we want a staircase to explain as much of the data as possible, and because MDL requires lossless compression, we need to encode both the cover of a staircase and the remainder of the data.

1) Encoding data inside the staircase: The first important observation to make is that a staircase consists of constant-row biclusters. Hence, we expect the data within each row in the staircase to be homogeneous, i.e. each row should consist of mostly the same (non-zero) symbols. The second observation is that if we encode the constant-row symbol S_r (see 2.4) for each row r, what remains is to encode the symbols that deviate from these *defaults*.

Since there are $|\sigma^+|$ possible default values, $\log|\sigma^+|$ bits are needed to encode a single default value. One default value needs to be encoded for each row that occurs in any of the biclusters in the staircase. The encoded size of all default values is then given by $L(S^{\text{defs}} \mid \mathcal{D})$:

$$L(S^{\text{defs}} \mid \mathcal{D}) = |rows(S)| \log|\sigma^+| \quad (17)$$

$$rows(S) = \{r \in M \mid \exists B_i \in S : r \in R_i\} \quad (18)$$

Since the row default values provide a default value for each position within the staircase, what remains is to encode all deviations from these defaults. Assuming that there are v (row,column) pairs within a staircase S, $\log v$ bits are required to uniquely identify one of these pairs. That is, we consider the concatenated vector of all values within $\mathcal{D}^S$ and index into this vector. For each value that deviates from the default, $\log(|\sigma|-1) = \log|\sigma^+|$ bits are needed to encode the deviating value. If there are u 'noisy' values, i.e. values that deviate from the default, encoding the data within the staircase requires $L(\mathcal{D}^S)$ bits, as follows:

$$L(\mathcal{D}^S) = \log v + u(\log v + \log|\sigma^+|) \quad (19)$$

The first term is required to encode u, the number of noisy values that follow after that.

2) Encoding any remaining data: For the remainder of the data, $\mathcal{D}^{\setminus S}$, we assume that the data is sparse, i.e. mostly zero, and all default values are hence 0. With this choice, staircases with larger covers are potentially rewarded with smaller compressed sizes.

Again, we encode only the values that deviate from the default, i.e. all non-zero values in this case. Given that $\mathcal{D}$ contains $m \times n$ values and $\mathcal{D}^S$ contains v values, we have that $\mathcal{D}^{\setminus S}$ consists of $w = (m \times n) - v$ values. Assuming that there are x non-zero values in $\mathcal{D}^{\setminus S}$, the encoded size for the remainder of the data is given by:

$$L(\mathcal{D}^{\setminus S}) = \log w + x(\log w + \log|\sigma^+|). \quad (20)$$

Putting everything together, we have that the encoded size of a dataset $\mathcal{D}$ encoded with a staircase S, denoted $L(\mathcal{D} \mid S)$, is defined as:

$$L(\mathcal{D} \mid S) = L(\mathcal{D}^{\text{defs}} \mid \mathcal{D}) + L(\mathcal{D}^S) + L(\mathcal{D}^{\setminus S}). \quad (21)$$

V. Experiments

In this section, we perform a number of experiments on both synthetic data and a real-world dataset to evaluate the proposed approach.

A. Data generation

We generated a synthetic data matrix of 1000 rows x 120 columns with two symbols in addition to the neutral zero, that is, $\sigma = \{0, g, r\}$. We sampled noise outside the staircase (background noise) from a multinomial distribution $\mu_{bn} = (\mu_1^{bg}, \mu_2^{bg}, \mu_3^{bg})$ where μ_1^{bg}, μ_2^{bg}, μ_3^{bg} are the probability of occurrences of g, r and 0 respectively. Within the data matrix, we planted a staircase of 500 rows x 100 columns with h bicluster, where h is a parameter. We sampled the value of a row in the staircase from a binomial distribution $\mu_h = (\mu_1^h, \mu_2^h)$, where μ_1^h and μ_2^h are the probabilities of rows being homogeneous in r and g respectively. We also sampled noise within the staircase from a binomial distribution $\mu_{sn} = (\mu_1^{sn}, \mu_2^{sn})$, where μ_1^{sn} and μ_2^{sn} are probability of non-zeroes (i.e. r or g) and zero.

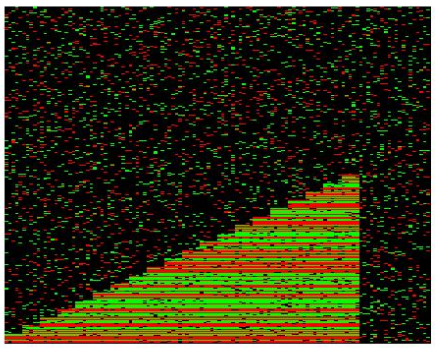


Fig. 3: A heatmap example of the synthetic data.

We did experiments with the following parameters, $\mu_{bn} = (0.05, 0.05, 0.90)$, $\mu_h = (0.7, 0.3)$ and $\mu_{sn} = (0.9, 0.1)$ and $h \in \{5, 10, 20\}$. Figure 3 shows a heatmap of a synthetic data with $h = 20$.

We used COMET [10] to implement the large neighbourhood search. Source code can be downloaded from our website, http://dtai.cs.kuleuven.be/CP4IM/staircase.

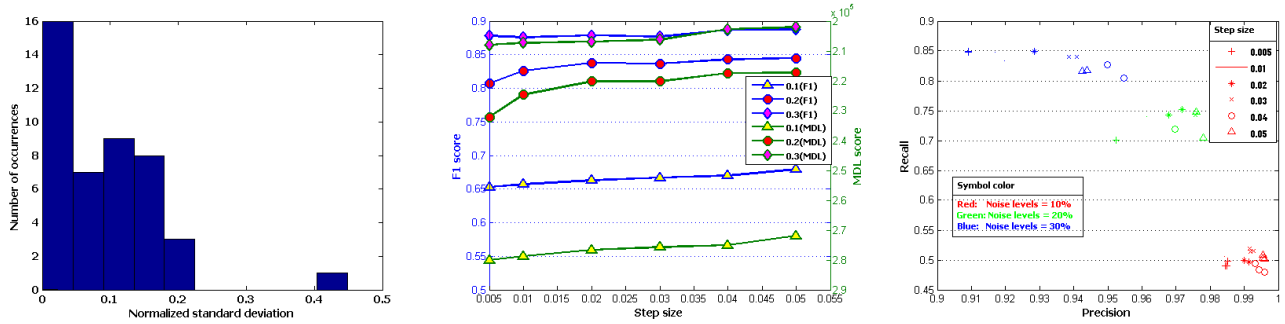
B. Experimental results with synthetic datasets

To evaluate the performance of the algorithm, we run it with varying values of the parameters (fault tolerance levels and step size θ). These experiments are used to answer the following questions:

1) How reliable is the local search procedure?: To this aim, we ran the staircase discovery algorithm on three synthetic datasets, using the data parameters given above. On each dataset, we ran the algorithm with different parameters (fault-tolerance and step-size) as well, and 6 times per parameter combination.

We evaluate how the MDL score varies among the runs by calculating its normalized standard deviation. We first calculate the standard deviation of the MDL score for the runs having the same parameter combination and then normalize it by the mean value. Figure 4a is the histogram of the normalized standard deviation of the MDL score. The result shows that most of the values concentrate nearby zero illustrating that the variation is typically low. In subsequent experiments we hence use three number of runs.

2) How reasonable is the MDL score?: In this experiment we consider the dataset with $h = 20$ biclusters. We vary the choices of step size θ ranging from 0.005 to 0.5 and fault-tolerance levels (shared between rows and columns) in the range of 0.10 to 0.30. For each solution found, we calculate our unsupervised MDL score. We also measure how well the



(a) Normalized standard deviation in MDL scores (b) Relating F1-Scores to MDL scores (c) Sensitivity analysis

Fig. 4: Experimental results

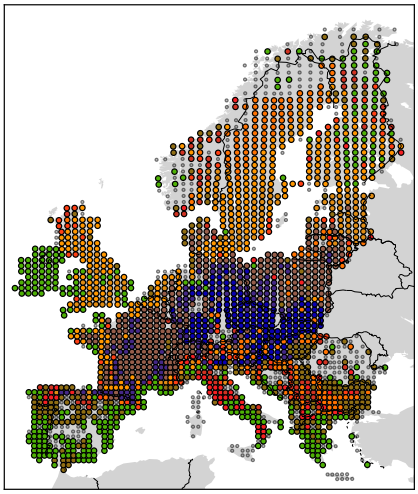


Fig. 5: Visualization of a staircase of mammals in Europe

staircases found correspond to the implanted staircase, using the F1 score. We expect that when the MDL score is low (lower is better), the F1 score reaches a high value. This is confirmed in Figure 4b (note that the right Y-axis, representing the MDL score, is inverted for readability). Unfortunately, the MDL score does not select the right number of biclusters in the implanted staircase.

3) How sensitive is the mining model to the parameter values?: Figure 4c shows precision and recall for the same data and parameter settings used in the previous experiment. As expected, the algorithm has different performance depending on the combination of parameter values. With less noise thresholds, for example 10% as illustrated in this experiment, most of the area that the algorithm predicts part of the true implanted staircase is correct (high precision) but it recovers a small part of the true staircase (low recall). With an appropriate choice of parameter combination, we could have both precision and recall high as in the case of using 30% of noise thresholds in this experiment. In practice, we have to do many experiments with different parameter values in order to find the approximate optimal solution. This is the weakness of the current method which can be a room for further improvement.

C. Experimental results with real dataset

We used the Mammals dataset [9], which consists of presence information of European mammals [14] and climate information for areas of 50×50 kilometres. Rows in this dataset correspond to regions, while columns correspond to species.

We run our staircase discovery algorithm on the Mammals dataset. The results are visualized in Figure 5, where blue indicates the last step in the staircase (hence, a region with many different species) and green indicates regions that strictly belong to the first step in the staircase (regions with few different species). The visualization shows that there is a trend of living of species in Europe. There are some small number of species can live in many areas of Europe whereas there are some particular areas that can house a larger number of species. For example, the green color scatters across the map (relating the large number of areas where a small number of species can live) while the blue color concentrates in the center (relating the area where a larger number of species can live).

VI. Related work

The problem of biclustering was first introduced in 2000 by the work of [4]. Many solutions have been proposed, to name just a few, (non-parametric) probabilistic inference [17], [19], statistical relational models [20]. However, these studies did not pay attention to the overlapping structure of the detected biclusters as our work. Frequent itemset mining, i.e. the discovery of subsets of items in transactional databases, has been applied in a relatively small number of biclustering studies as well. Examples include research by Serin and Vingron, and Blachon et al. [3], [16]. They deal with noise through a post-processing step; our proposed approach has a direct way to describe fault-tolerant itemsets in the framework of CP. Directly mining fault-tolerant patterns has been studied by Besson et al., also in a constraint programming setting [1], [2]. In our work, the fault-tolerant frequent itemset formalism is inspired by the one proposed by Liu et al. [11], which also imposes relative error constraints on rows and columns. However, Liu et al. do not consider the problem of finding staircases.

This works fits within the more general research area of finding pattern sets. Pattern set discovery has been studied in

a constraint programming setting before [8]; and MDL has successfully been applied in pattern set mining as well [18].

Conceptually, the structures in data that we are looking for are similar to nested segments [13]. However, in contrast to this work of Mannila and Terzi, the structure we propose to identify does not necessarily have to hold in the complete data. Finally, the structure we identify is related to that of a banded matrix [6]; the difference is that a banded matrix is sparse both above and below an (imaginary) diagonal identified in the data.

VII. Conclusions

In this paper we introduced the problem of finding staircases in data. Our method is based on a strict separation between the generation of candidate staircases and their evaluation in terms of the MDL score. Experiments show that the solutions having low MDL scores have high F1 scores and that the MDL score finds an implanted staircase with good precision and recall on artificial data.

There are several possibilities for future work. First, our current approach only identifies one staircase. Second, the two-phased approach taken in this work requires a large number of parametrized constraints; determining the right parameter values is computationally demanding. To address this problem, we are exploring approaches in which the MDL score and the constraints used to identify the staircases are more tightly integrated. Finally, there is room for improving the MDL score such that better models are selected.

Acknowledgment

This research was supported by the DBOF 10/044 Project, the Natural and Artifical Genetic Variation in Microbes project, a Postdoc grant for SN and project "Principles of Patternset Mining" by the Research Foundation – Flanders, a Rubicon grant for MvL by the Netherlands Organisation for Scientific Research, and the EU FET Open project "Inductive Constraint Programming".

References

[1] J. Besson, J-F. Boulicaut, T. Guns, S. Nijssen, Generalizing Itemset Mining in a Constraint Programming Setting. *Inductive Databases and Constraint-Based Data Mining*, 2010, 107-126.

[2] J. Besson, R. Pensa, C. Robardet, J-F. Boulicaut, Constraint-based mining of fault-tolerant patterns from Boolean data. *Knowledge Discovery in Inductive Databases*, 2005, 55-71.

[3] S. Blachon, R. Pensa, J. Besson, C. Robardet, J-F. Boulicaut, O. Gandrillon, Clustering formal concepts to discover biologically relevant knowledge from gene expression data. *In Silico Biology*, 2007, 7:0033.

[4] Y. Cheng, G.M. Church, Biclustering of expression data, *International Conference on Intelligent Systems for Molecular Biology*, 2000, 8:93-103.

[5] Q. Fu, A. Banerjee, Bayesian overlapping subspace clustering, *ICDM*, 2009.

[6] G.C. Garriga, E. Junttila, H. Mannila, Banded structure in binary matrices, *Knowl. Inf. Syst.*, 2011, 28(1), 197-226.

[7] T. Guns, S. Nijssen, L. De Raedt, Itemset Mining: a Constraint Programming Perspective, *Artif. Intell.*, (2011) 1951-1983.

[8] T. Guns, S. Nijssen, A. Zimmermann, L. De Raedt, Declarative heuristic search for pattern set mining, *Declarative Pattern Mining*, 2011.

[9] H. Heikinheimo, M. Fortelius, J. Eronen,H. Mannila Biogeography of european land mammals shows environmentally distinct and spatially coherent clusters, *J Biogeogr* 34(6):10531064, 2007.

[10] P. Van Hentenryck and L. Michel, *Constraint-based local search*, MIT Press, 2005.

[11] J. Liu, S. Paulsen, W. Wang, A. Nobel, J. Prins, Mining Approximate Frequent Itemsets from Noisy Data, *International Conference on Data Mining*, 2005, 721-724.

[12] S.C. Maderia, A.L. Oliveira, Biclustering algorithms for biological data analysis: a survey, *IEEE/ACM Trans Comput Biol Bioinform* 2004, 1:24-45.

[13] H. Mannila, E. Terzi, Nestedness and segmented nestedness, *Knowledge Discovery in Databases*, 2007: 480-489.

[14] AJ Mitchell-Jones, G. Amori, W. Bogdanowicz, B. Krystufek, PJH Reijnders, F. Spitzenberger, M. Stubbe, JBM Thissen, V. Vohralik, J. Zima, *The atlas of european mammals*. Academic Press, London, 1999.

[15] S. Nijssen, J. Vreeken, A. Zimmermann, N. Tatti, B. Bringmann, Mining Sets of Patterns – Next Generation Pattern Mining, *Tutorial at the International Conference on Data Mining*, 2011.

[16] A. Serin, M. Vingron, DeBi: Discovering Differentially Expressed Biclusters using a Frequent Itemset Approach, *Algorithms for Molecular Biology* 2011, 6:18.

[17] H. Shan, A. Banerjee, Bayesian Co-clustering, *IEEE International Conference on Data Mining*, 2008.

[18] J. Vreeken, M. van Leeuwen, A. Siebes, Krimp: mining itemsets that compress, *Data Mining and Knowledge Discovery*, 2011, 23(1), 169-214.

[19] P. Wang, K.B. Laskey, C. Domeniconi, M. Jordan, Nonparametric Bayesian Co-clustering Ensembles, in *Proceedings of the SIAM International Conference on Data Mining*, Mesa, Arizona, April 28-30, 2011.

[20] H. Zhao, L. Cloots, T.V.D. Bulcke, Y. Wu, R. De Smet, V. Storms, P. Meysman, K. Engelen and K. Marchal, Query-based biclustering of gene expression data using Probabilistic Relational Models, *BMC Bioinformatics* 12(Suppl 1): S37, 2011.

Appendix

Theorem A.1 (Fault tolerant cover constraint): The fault-tolerant cover set $\varphi^{ft}_{\mathcal{D}}(C, \epsilon_R)$ can be derived from T if the following constraint is satisfied:

$$\forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} (\mathcal{D}'_{t,i} - \epsilon_R) I_i \geq 0, \tag{22}$$

and equals

$$\varphi^{ft}_{\mathcal{D}}(C, \epsilon_R) = \{r \in M \mid \exists x \in \{1, \ldots |\sigma^+|\} : T_{r|\sigma^+|+x} = 1\}.$$

Proof:

$$\forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} (\mathcal{D}'_{t,i} - \epsilon_R) I_i \geq 0 \tag{23}$$

$$\Longleftrightarrow \forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} \mathcal{D}'_{t,i} I_i \geq \epsilon_R \sum_{i \in \mathcal{I}} I_i \tag{24}$$

using $I_i = 1 \leftrightarrow i \in C$ and $\sum_{i \in \mathcal{I}} I_i = \sum_{i \in \mathcal{I}} [i \in C] = |C|$:

$$\forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \sum_{i \in \mathcal{I}} \mathcal{D}'_{t,i} [i \in C] \geq \epsilon_R |C| \tag{25}$$

$$\Longleftrightarrow \forall t \in \mathcal{T} : T_t = 1 \leftrightarrow \frac{1}{|C|} \sum_{i \in C} \mathcal{D}'_{t,i} \geq \epsilon_R \tag{26}$$

using $\mathcal{D}'_{r',c} = 1 \leftrightarrow \mathcal{D}_{r,c} = {\sigma^+}_x$ for $r' = (r * |\sigma^+| + x)$:

$$\forall r \in M, \forall x \in \{1, \ldots, |\sigma^+|\} : \quad T_{(r*|\sigma^+|+x)} = 1 \leftrightarrow \frac{1}{|C|} \sum_{i \in C} [\mathcal{D}_{r,i} = {\sigma^+}_x] \geq \epsilon_R. \tag{27}$$

Going back to Definition 2.3 (Equation (2)) we have that:

$$r \in \varphi^{ft}_{\mathcal{D}}(C, \epsilon_R) \leftrightarrow \exists s \in \sigma^+ : \frac{1}{|C|} \sum_{c \in C} [\mathcal{D}_{r,c} = s] \geq \epsilon_R \tag{28}$$

$$\Longleftrightarrow r \in \varphi^{ft}_{\mathcal{D}}(C, \epsilon_R) \leftrightarrow \exists x \in \{1, \ldots, |\sigma^+|\} : T_{(r*|\sigma^+|+x)} = 1 \tag{29}$$

which concludes the proof. ∎