

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/317171985>

The PRIMING routine—Tiling through proximal alternating linearized minimization

Article in *Data Mining and Knowledge Discovery* · April 2017

DOI: 10.1007/s10618-017-0508-z

CITATIONS

4

READS

35

3 authors:



Sibylle Hess

Technische Universität Dortmund

5 PUBLICATIONS 6 CITATIONS

[SEE PROFILE](#)



Katharina Morik

Technische Universität Dortmund

235 PUBLICATIONS 2,271 CITATIONS

[SEE PROFILE](#)



Nico Piatkowski

Technische Universität Dortmund

31 PUBLICATIONS 196 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Rank Selection in Matrix Factorization [View project](#)



SFB-876 Project C3 [View project](#)

The PRIMPing Routine — Tiling through Proximal Alternating Linearized Minimization

Sibylle Hess · Katharina Morik ·
Nico Piatkowski

Received: date / Accepted: date

Abstract Mining and exploring databases should provide users with knowledge and new insights. Tiles of data strive to unveil true underlying structure and distinguish valuable information from various kinds of noise. We propose a novel Boolean matrix factorization algorithm to solve the tiling problem, based on recent results from optimization theory. In contrast to existing work, the new algorithm minimizes the description length of the resulting factorization. This approach is well known for model selection and data compression, but not for finding suitable factorizations via numerical optimization. We demonstrate the superior robustness of the new approach in the presence of several kinds of noise and types of underlying structure. Moreover, our general framework can work with any cost measure having a suitable real-valued relaxation. Thereby, no convexity assumptions have to be met. The experimental results on synthetic data and image data show that the new method identifies interpretable patterns which explain the data almost always better than the competing algorithms.

Keywords Tiling · Boolean Matrix Factorization · Minimum Description Length principle · Proximal Alternating Linearized Minimization · Nonconvex-Nonsmooth Minimization · Alternating Minimization

1 Introduction

In a large range of data mining tasks such as Market Basket Analysis, Text Mining, Collaborative Filtering or DNA Expression Analysis, we are interested in the exploration of data which is represented by a binary matrix. Data exploration is unsupervised by nature; the objective is to gain insight by a summarization of its relevant parts. Here, we seek for sets of columns and rows whose intersecting positions frequently feature a one. This identifies, e.g., groups of users together with their shared preferences, genes that are often co-expressed among several tissue samples, or words that occur together in documents describing the same topic. The identification of such sets of columns and rows is studied from the perspective of

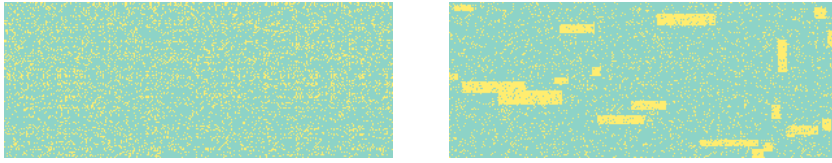


Fig. 1: Example binary dataset (left) with ones in yellow and zeros in blue. A rearrangement of columns and rows unveils structure (right).

various data mining subfields as *biclustering*, *tiling* or *matrix factorization* (Tatti and Vreeken 2012; Zimek and Vreeken 2013).

Consider the example binary database presented on the left in Fig. 1. The distribution of ones appears disarrayed, but a suitable permutation of columns and rows reveals a formation of blocks, depicted by the matrix on the right. Interpreting all binary entries which do not fit to this formation as noise, we aim to separate the haphazard component from the formative one.

This objective is difficult to delineate: where to draw the line between structure and noise? Are there natural limitations on the amount of blocks to derive? To what extent may they overlap? Miettinen and Vreeken (2014) successfully apply the *Minimum Description Length* (MDL) principle to reduce these considerations into one objective: exploit just as many regularities as serves the compression of the data. Identifying regularities with column-row interrelations, the description length counterbalances the complexity of the model (derived interrelations) and the fit to the data, measured by the size of the encoded data using the model. Decisive for the feasibility of extracted components is the definition of the encoding.

Miettinen and Vreeken (2014) evaluate several encodings with respect to their ability to filter a planted structure from noise. The method they use applies a greedy Boolean matrix factorization to extract candidate interrelations which are selected according to a specified description length. Another framework proposed by Lucchese et al (2014) greedily selects the interrelations directly in accordance with the description length. Most recently, Karaev et al (2015) propose another greedy algorithm with focus on a setting where ones more probably indicate interrelations than noise. All these methods are capable to identify the underlying structure in respectively examined settings. All in all, the experiments indicate however that the quality considerably varies depending on the distribution of noise and characteristics of the dataset (Miettinen and Vreeken 2014; Karaev et al 2015).

For real-world datasets, it is difficult (if not impossible) to estimate these aspects, in order to choose the appropriate algorithm or to assess its quality on the given dataset. Believing that the unsteady performance is due to a lack of theoretical foundation, we introduce a framework called *PAL-Tiling* to numerically optimize a cost measure which has a suitable real-valued approximation. In this respect, we derive approximations of two MDL cost measures, consequently proposing two algorithms: one applying $L1$ -regularization on the matrix factorization (PANPAL) and one employing an encoding by code tables as proposed by Siebes et al (2006) (PRIMP). We assess the algorithms' ability to filter the *true* underlying structure from the noise. Therefore, we compare various performance measures in a controlled setting of synthetically generated data as well as for real-

world data. We show that PRIMP is capable of recovering the latent structure in spite of varying database characteristics and noise distributions. In addition, we visualize the derived categorization into tiles by means of images, showing that our conducted minimization procedure of *PAL-Tiling* yields interpretable groupings.

1.1 Roadmap

In Section 2 we introduce our notation and review the work related to the three research branches of Tiling, MDL, and Nonnegative Matrix Factorization, which compound our method. After surveying these building blocks, we introduce our optimization framework PAL-TILING in Sec. 3. We derive the proximal mapping with respect to the proposed penalization of non-binary values, enabling the minimization of the approximate Boolean matrix factorization error under convergence guarantees. Therewith we derive the $L1$ -regularized minimization of the reconstruction error by the algorithm PANPAL in Sec. 3.3. We formulate the encoding via code tables in the form of a Boolean matrix factorization, which defines together with a suitable relaxation of this measure, derived in Sec. 3.4, the algorithm PRIMP. In Sec. 4, we compare our approach to related methods in various synthetically generated settings and real-world data. Furthermore, we provide insight into the algorithms' understanding of noise based on images. Finally, we conclude in Sec. 5.

2 Problem Definition and Building Blocks

We identify items $\mathcal{I} = \{1, \dots, n\}$ and transactions $\mathcal{T} = \{1, \dots, m\}$ by a set of indices of a binary matrix $D \in \{0, 1\}^{m \times n}$. This matrix represents the data, having $D_{ji} = 1$ iff transaction j contains item i . A set of items is called a *pattern*. If the pattern is a subset of a transaction, we say the transaction *supports* the pattern.

Throughout the paper, we often employ the function θ_t which rounds real to binary values, i.e., $\theta_t(x) = 1$ for $x \geq t$ and $\theta_t(x) = 0$ otherwise. We abbreviate $\theta_{0.5}$ to θ and denote with $\theta(X) = (\theta(X_{ji}))_{ji}$ the entry-wise application of θ to a matrix X .

We denote matrix norms as $\|\cdot\|$ for the Frobenius norm and $|\cdot|$ for the entry-wise 1-norm. These norms are equivalent for binary matrices X in the sense that $|X| = \|X\|^2$. We use the short notation $|X|_- = |\theta(-X)|$ and $|X|_+ = |\theta(X)|$ to separate the norm of negative and nonnegative entries of X . We often abbreviate the notation of a matrix $(x_{ij})_{1 \leq i \leq n, 1 \leq j \leq m}$ to $(x_{ij})_{ij}$ if the range of indices is clear from the context. Correspondingly, we notate column vectors $(x_i)_i$. The operator \circ denotes the Hadamard product which multiplies two matrices of same dimensions element-wise. Lastly, we remark that \log denotes the natural logarithm.

2.1 Problem Definition

We seek sets of column-row selections which can be visualized as a formation of blocks as exemplified in our introduction. The expectation that such a formation exists is based on the assumption that the data D originates from a Boolean

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} = \theta \left(\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \right)$$

Fig. 2: An exact Boolean factorization using two tiles. The tiles are highlighted.

matrix product $\theta(YX^T)$. Here, it is important to understand that the thresholding function θ (defined above) suffices to map binary operations onto the Boolean algebra where the addition corresponds to the logical conjunction, i.e., $\theta(0+1) = \theta(1+0) = 1$ and $\theta(1+1) = 1$. This way, the product is also well-defined for nonnegative real-valued matrices. For an exploration of non-canonical Boolean matrix products, and what derives from them, see, e.g., (Miettinen 2015).

Let X be an $n \times r$ binary matrix and Y an $m \times r$ binary matrix. The product $\theta(YX^T)$ specifies r column-row selections called tiles, one by each pair of column vectors $(X_{\cdot s}, Y_{\cdot s})$. Thereby, we implicitly assume that each tile provides information about co-occurrences of items and transactions. In practice, however, one column vector may be equal to the zero vector or indicate only one item, respectively a single usage of a pattern. We do not take these trivial column-row selections into account and introduce the function $r(\cdot, \cdot)$ to count the number of valuable tiles $r(X, Y) = |\{s : |X_{\cdot s}| > 1 \wedge |Y_{\cdot s}| > 1\}| \leq r$. In theory, we often assume that $r = r(X, Y)$ and in this case, we call r the rank of the tiling or factorization. An example of a rank-2 factorization is depicted in Fig. 2. The vector $X_{\cdot s}$ indicates the pattern which contains all items $i \in \mathcal{I}$ with $X_{is} = 1$. Likewise, the vector $Y_{\cdot s}$ marks the transactions which use pattern $X_{\cdot s}$ to form the tile. Accordingly, we refer to the matrix X as the *pattern matrix* and to Y as the *usage matrix*. The name *tile* reflects its visualization as a single block matrix $Y_{\cdot s} X_{\cdot s}^T$ for suitably rearranged columns and rows.

We state the informal problem to recover the latent factorization, which we intend to approach in this paper, as follows.

Informal Problem Definition

Given a data matrix $D \in \{0, 1\}^{m \times n}$ originating from the following process

1. Let $X \in \{0, 1\}^{n \times r}$ be a rank r matrix, denoting r patterns.
2. For each transaction D_j , choose a set of patterns $S_j \subseteq \{1, \dots, r\}$.
3. Construct $Y \in \{0, 1\}^{m \times r}$ such that $Y_{js} = 1 \Leftrightarrow s \in S_j$.
4. Let $N \in \{-1, 0, 1\}^{m \times n}$ be the noise matrix satisfying

$$N_{ji} + \theta(YX^T)_{ji} \geq 0.$$

5. Set

$$D = \theta(YX^T) + N. \quad (1)$$

Find the factor matrices X and Y .

Algorithm	$f(X, Y, D)$	Constraints \mathcal{C} ($N = D - \theta(YX^T)$)	\mathcal{R}
LTM	r	$ N = 0$	\mathbb{N}
r -LTM	$- \theta(YX^T) $	$ N _- = 0$	$\{r\}$
HYPER+	$ X + Y $	$ N _+ = 0, N _- \leq \beta$	\mathbb{N}
MAXENT	$\frac{IC_p(X, Y)}{L_p(X, Y)}$	\emptyset	\mathbb{N}
ASSO	$f_{RSS}(X, Y, D)$	\emptyset	$\{r\}$
KRIMP, SLIM, SHRIMP	$f_{CT}(X, Y, D)$	$ N _- = 0$	\mathbb{N}
GROEI	$f_{CT}(X, Y, D)$	$ N _- = 0$	$\{r\}$
PANDA	$f_{LI}(X, Y, D)$	\emptyset	$\{r\}$
MDL4BMF, NASSAU	$f_{TXD}(X, Y, D)$	\emptyset	$\mathbb{N}_{\leq \min\{n, m\}}$
MDL4BMF, PANDA+	$f_{TX}(X, Y, D)$	\emptyset	$\mathbb{N}_{\leq \min\{n, m\}}$

Table 1: Overview of tiling cost measures and implementing algorithms. $\mathbb{N}_{\leq a}$ denotes the set of natural numbers less than or equal to a .

Solving this problem is infeasible in practice as the data generation process is not invertible. Therefore, an approachable surrogate problem is formulated: the minimization of a function, which shall, together with suitable constraints on the result space, indicate the quality of the derived model. In the following, we inspect how three related research branches formulate and tackle such minimization problems, namely *Tiling*, the *Minimum Description Length* principle and *Nonnegative Matrix Factorization*.

2.2 Tiling

Tiling addresses the task to find binary matrices which minimize a given cost measure in a restricted search space. Many cost measures have been formulated with respect to tiling. Each one defines different criteria of what makes a set of tiles suitable. We list the most important considerations in Table 1 according to the following task formalization:

Tiling

Given a binary database $D \in \{0, 1\}^{m \times n}$, a set of real-valued functions $c \in \mathcal{C}$, a set of natural numbers \mathcal{R} and a cost measure f .

Find a tiling

$$\begin{aligned} & \min_{(X, Y)} f(X, Y, D) \\ & \text{subject to } c(X, Y, D) \leq 0, \quad c \in \mathcal{C} \\ & \quad X \in \{0, 1\}^{n \times r}, \quad Y \in \{0, 1\}^{m \times r}, \quad r \in \mathcal{R} \end{aligned}$$

We see in Table 1 that many algorithms prohibit negative noise ($|N|_- = 0$). We call such tilings *restrained*, as the usage of patterns is restrained to the supporting transactions.

Geerts et al (2004) consider the cost measure as a measure of interestingness of patterns. In this setting, a tile is determined by its pattern because its usage is

identified with all supporting transactions. This implicitly excludes negative noise but enables the application of pattern-mining techniques in the algorithms LTM and r-LTM.

Kontonasios and De Bie (2010) and Xiang et al (2011) argue that the integration of negative noise enables more succinct descriptions and makes the tiling robust to noise. If negative noise is not allowed, every flip of a single bit in the interior of a tile breaks it into two. Xiang et al (2011) propose with the greedy algorithm HYPER+ to mine restrained tiles first and to combine them to larger (noisy) tiles in a second step, as long as a specified amount of negative noise is not exceeded. Kontonasios and De Bie (2010) propose the information theoretical regulation of noise. The algorithm MAXENT greedily selects the tile with the highest information ratio among a set of input candidate tiles. The information ratio puts the information content $IC_p(X, Y)$ in relation to the description length $L_p(X, Y)$ of a tiling, given a maximum entropy distribution p over data matrices. Both algorithms include negative noise only in a post-processing step and provide no mechanisms to directly derive suitable unrestrained tiles.

Miettinen et al (2008) strive for direct minimization of the approximation error under the umbrella term Boolean Matrix Factorization (BMF). They show that the tiling of rank r which yields the minimum error $f_{\text{RSS}}(X, Y, D) = |D - \theta(YX^T)|$ cannot be approximated within any factor in polynomial time (unless $\mathbf{NP} = \mathbf{P}$). Accordingly, they propose a heuristic to solve this problem. ASSO incrementally creates r tiles by selecting a pattern first and minimizing the error subject to the usage afterwards. Decisive for the quality of the returned factorization is the choice of the rank r . To determine this parameter automatically, several algorithms implement one key paradigm called Minimum Description Length (MDL).

2.3 The MDL Principle

MDL is introduced by Rissanen (1978) as an applicable version of Kolmogorov complexity (Li 1997; Grünwald 2007). The learning task to find the best model according to the MDL principle is given as follows:

Description Length Minimization

Given data D and a set of models \mathcal{M} .

Find a model $M \in \mathcal{M}$ for D which minimizes the description length

$$L(D, M) = L^D(D, M) + L^M(M),$$

where $L^D(D, M)$ denotes the compression size of the database in bits (using model M for the encoding) and $L^M(M)$ is the description size in bits of the model M itself.

Specifications of this task differ in the definition of the encoding which defines the set of models \mathcal{M} . Typical models for tilings are given by the factorizations which satisfy the constraints

$$\mathcal{M} = \{(X, Y) \in \{0, 1\}^{n \times r} \times \{0, 1\}^{m \times r} \mid c(X, Y, N) \leq 0 \forall c \in \mathcal{C}, r \in \mathcal{R}\}.$$

An encoding which is successfully applied in the area of pattern mining and which we discuss later in the context of tiling, uses code tables as proposed by Siebes et al

(2006). The code table assigns optimal prefix-free codes to a set of patterns, such that the code lengths can be calculated without realizations of actual codes. We imagine the code table two-columned: itemsets are listed on the left and assigned codes on the right. Such a dictionary from itemsets to code words can be applied to databases similarly as code words to natural language texts. However, the code usage is not as naturally defined as for words in a text. Patterns are not nicely separated by blanks and the possibilities to disassemble a transaction into patterns are numerous. Therefore, we require for every transaction the indication of its cover by patterns of the code table. This is modeled by a function $cover$, which partitions D_j into patterns of the code table.

Let $CT = \{(X_\sigma, C_\sigma) | 1 \leq \sigma \leq \tau\}$ be a code table of τ patterns X_σ together with their assigned codes C_σ . For any distribution P over a finite set Ω , an optimal set of prefix-free codes exists (Cover and Thomas 2006, Theorem 5.4.1) such that the number of required bits for the code of $x \in \Omega$ is approximately

$$L(code(x)) \approx -\log(P(x)).$$

Desiring that frequently used codes are shorter in size, Siebes et al (2006) introduce the function $usage$ that maps a pattern to the number of transactions which use it for their cover, i.e.,

$$usage(X_\sigma) = |\{X_\sigma \in cover(CT, D_j.) | j \in \mathcal{T}\}|.$$

The probability mass function over all itemsets X_σ in the code table is defined as

$$P(X_\sigma) = \frac{usage(X_\sigma)}{\sum_{1 \leq \rho \leq \tau} usage(X_\rho)}. \quad (2)$$

This implies that $L(C_\sigma) = -\log P(X_\sigma)$. The data matrix is encoded by a transaction-wise concatenation of codes, denoted by the cover, i.e., transaction D_j is encoded by a concatenation of codes C_σ with $X_\sigma \in cover(CT, D_j.)$. Therefore, code C_σ occurs $usage(X_\sigma)$ times in the encoded dataset. The size of the data description is thus computed by

$$L_{CT}^D(D, CT) = - \sum_{1 \leq \sigma \leq \tau} usage(X_\sigma) \cdot \log(P(X_\sigma)).$$

The description of the model, the code table, requires the declaration of codes C_σ and corresponding patterns X_σ . Code C_σ has a size of $-\log(P(X_\sigma))$. A pattern is described by concatenated standard codes of contained items. Standard codes arise from the code table consisting of singleton patterns only, where the usage of singleton $\{i\}$ for $i \in \mathcal{I}$ is equal to the frequency $|D_{.i}|$. In conclusion, the description size of the model is computed as

$$L_{CT}^M(CT) = - \sum_{\substack{1 \leq \sigma \leq \tau \\ usage(X_\sigma) > 0}} \left(\log(P(X_\sigma)) + \sum_{i \in X_\sigma} \log\left(\frac{|D_{.i}|}{|D|}\right) \right).$$

Note that the function L_{CT} originally uses the logarithm with base two. We implicitly reformulate this description length by substituting with the natural logarithm. This is equivalent to multiplying the function by a constant which is

negligible during minimization. In return, using the natural logarithm will shorten the derivations in Sec. 3.4.

Siebes et al (2006) use a heuristic cover function for the algorithm KRIMP which employs a specified, static order on patterns. The cover function greedily selects the next pattern in the order which covers items that are not covered yet. This way, covering patterns must neither overlap nor cover more items than stated by the transaction. KRIMP examines an input set of frequent patterns in another static order, adding a candidate pattern to the code table whenever that improves the compression size. Additionally, pruning methods are proposed to correct the selection of patterns in the code table.

SLIM (Smets and Vreeken 2012) differs in its candidate generation, which is dynamically implemented according to an estimated compression gain and dependent on the current code table. This strategy typically improves the compression size, but mainly reduces the amount of returned patterns. Still, the number of considered candidates is extremely large in comparison to those who are accepted. Time consumption is dominated by computing the usage for each evaluated candidate. SHRIMP (Hess et al 2014) exploits the indexing nature of trees in order to efficiently identify those parts of the database which are affected by an extension of the code table. Siebes and Kersten (2011) restrict with the algorithm GROEI the code table to a constant number of patterns. They resort to a heuristic beam search algorithm, but only for tiny datasets, the beam width parameter can be set to a level allowing a reasonably wide enough exploration of the search space, or else the run time explodes.

All these algorithms follow the heuristic cover definition of KRIMP which prohibits negative noise and tile overlap (we state the function f_{CT} and discuss the specific relationship between the proposed encoding by code tables and tiling in Sec. 3.4). Although the employment of code tables is originally motivated as a methodology to obtain concise and compressing descriptions of the data, the encoding is quite wordy in comparison to the output of unrestrained tiling algorithms, which we discuss in the following section. Nonetheless, attempts to revoke the restraints of the tiling are not known to us.

2.4 Merging MDL and Tiling

MDL's incorporated trade-off between model complexity and data fit is apt for the determination of the factorization rank. Algorithms which determine the rank according to the MDL principle implement a similar scheme, so far. The costs are identified with the description length and in every iteration, the rank is increased as long as this results in decreasing costs. For every considered rank, a factorization (tiling) method is invoked, which usually extends the result from the former iteration. The performance of this method depends on the choice of factorization and encoding which determines the description length.

Lucchese et al (2010) propose an encoding as it is known for sparse data representations, describing a matrix only by the positions of ones. Consequently, the model is described with $L^M((X, Y)) = |X| + |Y|$ bits and the data with $L^D(D, (X, Y)) = |D - \theta(YX^T)|$ bits, up to a multiplicative constant. The resulting cost function is denoted as $f_{L1}(X, Y, D) = |D - \theta(YX^T)| + |X| + |Y|$. The

algorithm PANDA uses a factorization method which adds a tile to the current tiling in a two stage heuristic, comparable to HYPER+.

Miettinen and Vreeken (2014) argue that the encoding used in PANDA is too coarse. They investigate multiple encodings, applying ASSO to incrementally increase the factorization rank. Their best-performing encoding is called *Typed XOR DtM* encoding. This is based upon the description of n -dimensional binary vectors by number and distribution of ones. We refer to the Typed XOR DtM description length as f_{TXD} and to the corresponding algorithm as MDL4BMF. The experimental evaluation suggests that MDL4BMF’s rank estimation is accurate in a setting with moderate noise, i.e., less than 15%, and moderate number of planted tiles, i.e., less than 15. It seems to have a tendency to underfit, as opposed to PANDA, which returns sometimes ten times more tiles than planted.

On the other hand, the framework of PANDA can be applied with an arbitrary cost measure. Lucchese et al (2014) enhance the algorithm PANDA to a faster version PANDA+ and evaluate the ability to detect a planted tiling in relation to different cost measures and algorithms. In their evaluation of synthetically generated datasets with less than 10% equally distributed noise, PANDA+ using Typed XOR costs f_{TX} is outperforming any other choice. The performance is explained with the objective of PANDA+’s factorization method, which aims at minimizing the costs, in contrast to ASSO, minimizing only the noise.

Another algorithm which tries to incorporate the direct optimization of the MDL-cost measure is NASSAU (Karaev et al 2015). Remarking that the formerly proposed algorithms do not reconsider tiles mined at previous iterations, NASSAU refines the whole tiling every few steps in relation to the cost measure. Still, the incorporated factorization method minimizes solely the factorization error. The experiments focus on a setting where negative noise is prevalent. In this case, differences to MDL4BMF are often hard to capture while NASSAU typically outperforms PANDA+.

2.5 Nonnegative Matrix Factorization

The Boolean factorization of Eq. (1) has a popular relative called Nonnegative Matrix Factorization (NMF). Given a nonnegative, real valued matrix $D \in \mathbb{R}_+^{m \times n}$ and a rank $r \in \mathbb{N}$, the goal is to recover nonnegative factors $X \in \mathbb{R}_+^{n \times r}$ and $Y \in \mathbb{R}_+^{m \times r}$ such that $YX^T \approx D$. To find the “correct” factorization, again, several objective functions and constraints are proposed. Most commonly, the residual sum of squares (RSS) is minimized

$$\min_{X,Y} F(X, Y) = \frac{1}{2} \left\| D - YX^T \right\|^2. \quad (3)$$

The function F is nonconvex, but convex in either X or Y , if the other argument is fixed. That makes it suitable for the *Gauss-Seidel* scheme, also known as *block-coordinate descent* or *alternating least squares*, an alternating minimization along one of the matrices while the other one is fixed. That is, a sequence (X_k, Y_k) is

created by

$$\begin{aligned} X_{k+1} &\in \arg \min_X F(X, Y_k) \\ Y_{k+1} &\in \arg \min_Y F(X_{k+1}, Y). \end{aligned} \tag{4}$$

However, finding a minimum in every iteration is computationally intensive. Thus, existing algorithms for NMF approximate the scheme of Eq. (4) in several ways (Wang and Zhang 2013). Often, the minimization step is replaced by a single gradient descent update.

NMF is originally introduced by Paatero and Tapper (1994) under the name Positive Matrix Factorization. It received much attention since the publication of the easily implementable multiplicative update algorithm by Lee and Seung (2001). Their intuitive explanation of coherence between the nonnegativity constraints and the resulting parts-based explanation of the data (Lee and Seung 1999), emphasizes the interpretability of the results.

Although initially the difference between NMF and clustering was emphasized (Lee and Seung 1999), further research affirms inherent clustering properties (Li and Ding 2006). In this context, columns of X equate cluster centroids and corresponding columns of Y indicate cluster membership tendencies. Restricting Y to a binary matrix makes the memberships definite and the orthogonality constraint $Y^T Y = I$ enforces unique cluster assignments. This factorization task is actually equivalent to k -means (Ding et al 2005, 2006; Bauckhage 2015). If the data matrix is binary, a binary factorization is also desirable, at least to get interpretable results for the cluster centroids (Li 2005). In this way, the factorization can be read as a clustering of items, or by using the transposed product, as a clustering of transactions. This is also known under the terms *biclustering*, *co-clustering* or *subspace clustering*.

To the best of our knowledge, Zhang et al (2007) are the only ones approaching the task of biclustering in conjunction with alternating minimization, the standard procedure to solve NMF. They propose two methods: the first one uses gradient descent updates with the longest step size preserving nonnegativity of the factor matrices and integrates the penalization of non-binary values into the minimization of the factorization error. As penalizing function, they choose the Mexican hat function $\omega(x) = \frac{1}{2}(x^2 - x)^2$. The second method is designed to find the threshold at which nonnegative factor matrices might be rounded best to binary matrices.

Although these methods have several drawbacks (the former lacks a convergence guarantee and the latter applies a costly backtracking linesearch), the results are very promising in comparison to common greedy biclustering algorithms. However, this branch of research is considered to be substantially different from its formulation in Boolean algebra (Miettinen and Vreeken 2014; Lucchese et al 2014). Indeed, the numerical optimization of the binary factorization is not easily adopted for multiplications in Boolean algebra $\theta(YX^T)$; θ has a point of discontinuity at 0.5. Equally, all proposed cost measures in Table 1 are not continuous for real valued matrices with entries in $[0, 1]$.

3 Merging Tiling, MDL, and NMF

We wish to find a way out of the greedy minimization of tiling cost measures and ask to which extent the theory behind popular NMF optimization methods may be applied to Boolean matrix factorizations. In conclusion, we propose an adaptation of the Gauss-Seidel method to minimize a suitable relaxation of tiling cost measures. Similar to the thresholding algorithm of Zhang et al (2007), the matrices are rounded according to the actual cost measure afterwards. Moreover, we incorporate the determination of the factorization rank, utilizing that the cost measure may select fewer tiles than offered.

With our ambition to adapt the alternating minimization for Boolean matrix factorization, we face two problems: First, as mentioned above, the use of Boolean algebra induces points of discontinuity. In particular, the gradient of the cost measures does not exist at all points which hinders the application of standard gradient descent methods. Second, many tiling cost measures are not convex in X or Y , not even if the other argument is fixed, which is a necessary condition to prove the convergence of the Gauss-Seidel scheme.

To begin with, we inspect how NMF (Eq. (3)) and BMF deal with overlapping tiles. This is the crucial point where Boolean algebra diverges from elementary algebra. An illustration of a binary data matrix D consisting of two overlapping tiles and its approximation by a NMF is shown in the top two equations of Fig. 3. We see that the factors contain values smaller than one at entries which are involved in overlapping parts. With this, overlapping sections are equally well approximated as non-overlapping components. The matrices D_A and D_B in Fig. 3 show the resulting approximations when the nonnegative factor matrices are rounded to binary matrices. We find that the reconstruction error is largest when the binary matrices are multiplied in elementary algebra (matrix D_A in Fig. 3). This illustrates how binary matrix factorization penalizes overlapping patterns, a feature which is desirable in clustering when clusters are not allowed to overlap. Similarly, NMF would return less overlapping factors at a higher factorization rank. In this case, however, the original data matrix is exactly reconstructed by the Boolean product of thresholded factor matrices (matrix D_B in Fig. 3). That is why we consider the minimization of a relaxed cost measure with respect to the elementary algebra whereby the factorization rank is increased stepwise. An evaluation of the actual cost measure in Boolean algebra on the rounded matrices decides whether the rank shall be increased or not.

This leads us to the second concern, the minimization of a possibly not even partially convex objective. Bolte et al (2014) extend the application of the Gauss-Seidel scheme to such a larger class of functions with the *Proximal Alternating Linearized Minimization* (PALM). This technique focuses on objective functions which break down into a smooth part $F : \mathbb{R}^{n \times r} \times \mathbb{R}^{m \times r} \times \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ and a nonsmooth component $\phi : \{X \in \mathbb{R}^{m \times n} | m, n \in \mathbb{N}\} \rightarrow (-\infty, \infty]$

$$F(X, Y, D) + \phi(X) + \phi(Y). \quad (5)$$

Thereby, no convexity assumptions are made on F and ϕ . Furthermore, the function ϕ may return ∞ , which can be used to model restrictions of the search space, e.g., the non-negativity constraint of NMF. The method performs an alternating minimization on the linearized objective, substituting F with its first order Taylor

$$\begin{aligned}
D &= \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \approx \begin{pmatrix} 1 & .9 & .9 & .1 \\ .7 & 1.2 & 1.2 & .7 \\ .1 & .9 & .9 & 1 \end{pmatrix} \\
&\approx \begin{pmatrix} 1 & 0 \\ .6 & .6 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & .9 & .9 & .1 \\ .1 & .9 & .9 & 1 \end{pmatrix} \\
D_A &= \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} = \theta \begin{pmatrix} 1 & 0 \\ .6 & .6 \\ 0 & 1 \end{pmatrix} \cdot \theta \begin{pmatrix} 1 & .9 & .9 & .1 \\ .1 & .9 & .9 & 1 \end{pmatrix} \\
D_B &= \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} = \theta \left(\theta \begin{pmatrix} 1 & 0 \\ .6 & .6 \\ 0 & 1 \end{pmatrix} \cdot \theta \begin{pmatrix} 1 & .9 & .9 & .1 \\ .1 & .9 & .9 & 1 \end{pmatrix} \right)
\end{aligned}$$

Fig. 3: Approximation of a binary matrix D with two overlapping tiles (top) applying NMF (second from above) and the factorizations resulting from thresholding the factor matrices to binary matrices in elementary algebra (second from below) and Boolean algebra (below). Tiles are highlighted.

approximation. This is achieved by alternating *proximal mappings* from the gradient descent update with respect to F , i.e., the following steps are repeated for $1 \leq k \leq K$:

$$X_{k+1} = \text{prox}_{\alpha_k \phi}(X_k - \alpha_k \nabla_X F(X_k, Y_k, D)); \quad (6)$$

$$Y_{k+1} = \text{prox}_{\beta_k \phi}(Y_k - \beta_k \nabla_Y F(X_{k+1}, Y_k, D)). \quad (7)$$

The proximal mapping of ϕ , $\text{prox}_\phi : \text{dom}(\phi) \rightarrow \text{dom}(\phi)^1$ is a function which returns a matrix satisfying the following minimization criterion:

$$\text{prox}_\phi(X) \in \arg \min_{X^*} \left\{ \frac{1}{2} \|X - X^*\|^2 + \phi(X^*) \right\}.$$

Loosely speaking, the proximal mapping gives its argument a little push into a direction which minimizes ϕ . For a detailed discussion, see, e.g., (Parikh and Boyd 2014). As we can see in Eqs. (6) and (7), the evaluation of this operator is a base operation. Similarly to the alternating minimization in Eq. (4), finding the minimum of the proximal mapping in every iteration by numerical methods is infeasible in practice. Thus, the trick is to use only simple functions ϕ for which the proximal mapping can be calculated in a closed form.

The variables α_k and β_k in Eqs. (6) and (7) are the step sizes, which are computed under the assumption that the partial gradients $\nabla_X F$ and $\nabla_Y F$ are globally Lipschitz continuous with moduli $M_{\nabla_X F}(Y)$ and $M_{\nabla_Y F}(X)$, i.e.,

$$\|\nabla_X F(X_1, Y, D) - \nabla_X F(X_2, Y, D)\| \leq M_{\nabla_X F}(Y) \|X_1 - X_2\|$$

¹ $\text{dom}(\phi)$ is the domain of ϕ

for all $X_1, X_2 \in \mathbb{R}^{n \times r}$ and similarly for $\nabla_Y F$. If F computes the RSS as stated in Eq. (3), the Lipschitz moduli are given as

$$M_{\nabla_X F}(Y) = \|YY^T\|, \quad M_{\nabla_Y F}(X) = \|XX^T\|.$$

The step sizes are computed as

$$\alpha_k = \frac{1}{\gamma M_{\nabla_X F}(Y_k)}, \quad \beta_k = \frac{1}{\gamma M_{\nabla_Y F}(X_{k+1})},$$

where γ is a constant larger than one. The parameter γ ensures that the step size is indeed smaller than the inverse Lipschitz constant, which is required to guarantee the convergence. Note that the step sizes are antimonotonic to γ , i.e., if $\gamma = 2$ then the step sizes are almost half as small as they could be. Assuming that the infimum of F and ϕ exists and ϕ is proper and lower continuous, PALM generates a nonincreasing sequence of function values which converges to a critical point.

3.1 PAL-Tiling: General Framework

We employ the optimization scheme PALM to minimize a specified relaxation of a tiling cost measure. Adapting to the terminology of Eq. (5), we assume that for a factor $\mu \geq 0$ and regularizing function G the relaxation has the form

$$F(X, Y, D) = \frac{\mu}{2} \|D - YX^T\|^2 + \frac{1}{2} G(X, Y). \quad (8)$$

Here the multiplication by one half refers to the traditional formulation of the residual sum of squares in Eq. (3), which shortens the formulation of gradients. The regularizing function G is supposed to be real valued and smooth with partial gradients which are Lipschitz-continuous with moduli $M_{\nabla_Y G}(X)$ and $M_{\nabla_X G}(Y)$. That is,

$$\|\nabla_X G(X_1, Y) - \nabla_X G(X_2, Y)\| \leq M_{\nabla_X G}(Y) \|X_1 - X_2\|,$$

and similarly for $\nabla_Y G$. It follows from the triangle inequality that the Lipschitz moduli of the partial gradients of F are given by the sum

$$\begin{aligned} M_{\nabla_X F}(Y) &= \mu \|YY^T\| + \frac{1}{2} M_{\nabla_X G}(Y) \\ M_{\nabla_Y F}(X) &= \mu \|XX^T\| + \frac{1}{2} M_{\nabla_Y G}(X). \end{aligned}$$

We use the function ϕ , which is integrated into the objective function as stated in Eq. (5), to limit the matrix entries to the interval $[0, 1]$, i.e., $X \in [0, 1]^{n \times r}$ and $Y \in [0, 1]^{m \times r}$. As discussed by Zhang et al (2007), this prevents an imbalance between the factor matrices in which one matrix is very sparse and the other very dense. Apart from that, we wish that the relaxed optimization returns factor matrices which are as close to binary matrices as possible. Therefore, we incorporate penalty terms for non-binary matrix entries in the function ϕ . Choosing ϕ as a proper and lower semicontinuous function (to be defined in Sec. 3.2), the objective meets the requirements of PALM to guarantee the convergence to a critical point in a nonincreasing sequence of iterative function values.

Algorithm 1 Proximal Alternating Linearized Tiling

```

1: function PAL-TILING( $D, \Delta_r, K, T, \gamma = 1.00001$ )
2:    $(X_K, Y_K) \leftarrow (\emptyset, \emptyset)$ 
3:   for  $r \in \{\Delta_r, 2\Delta_r, 3\Delta_r, \dots\}$  do
4:      $(X_0, Y_0) \leftarrow \text{INCREASERANK}(X_K, Y_K, \Delta_r)$  ▷ Append  $\Delta_r$  random columns
5:     for  $k \in \{0, \dots, K-1\}$  do
6:        $\alpha_k^{-1} \leftarrow \gamma M_{\nabla_X F}(Y_k)$ 
7:        $X_{k+1} \leftarrow \text{prox}_{\alpha_k \phi}(X_k - \alpha_k \nabla_X F(X_k, Y_k, D))$ 
8:        $\beta_k^{-1} \leftarrow \gamma M_{\nabla_Y F}(X_{k+1})$ 
9:        $Y_{k+1} \leftarrow \text{prox}_{\beta_k \phi}(Y_k - \beta_k \nabla_Y F(X_{k+1}, Y_k, D))$ 
10:    end for
11:     $(X, Y) \leftarrow \arg \min\{f(\theta_x(X_K), \theta_y(Y_K)) \mid x, y \in T\}$  ▷ Threshold to binary matrices
12:    if  $r - r(X, Y) > 1$  then
13:      return  $(X, Y)$ 
14:    end if
15:  end for
16: end function

```

We sketch our method, Proximal Alternating Linearized Tiling (PAL-TILING), in Algorithm 1. A data matrix D , rank increment Δ_r , maximum number of iterations K , a set of threshold values T and the parameter γ , having a default value of $\gamma = 1.00001$, are the input of this algorithm. For every considered rank, we perform the proximal alternating linearized minimization of the relaxed objective (line 5-10). After the numerical minimization of the relaxed objective F , the matrices X_K and Y_K , having entries between zero and one, are rounded to binary matrices X and Y with respect to the actual cost measure f (line 11). If the rounding procedure returns binary matrices which use at least one (non-singleton) pattern less than possible, the current factorization is returned. Otherwise, we increase the rank and add Δ_r random columns with entries between zero and one to the relaxed solution of the former iteration (X_K, Y_K) .

To apply this scheme, we need to define the penalizing function ϕ , derive its proximal mapping in a closed form and find a suitable cost measure with its smooth relaxed approximation.

3.2 Penalizing Non-Binary Values

While the Mexican hat function can be seen as an L_2 regularization equivalent penalizer for binary values, here, we choose an L_1 -equivalent form. Specifically, we choose $\phi(X) = \sum_{i,j} \Lambda(X_{ij})$, which employs the one-dimensional function

$$\Lambda(x) = \begin{cases} -|1 - 2x| + 1 & x \in [0, 1] \\ \infty & \text{otherwise.} \end{cases}$$

to restrict matrix entries to the interval $[0, 1]$ and to penalize non-binary values. The curve of Λ is depicted in Fig. 4. We derive with the following proposition a closed form for the computation of the exact minimum as assigned by the proximal mapping with respect to ϕ .

Theorem 1 *Let $\alpha > 0$ and $\phi(X) = \sum_{i,j} \Lambda(X_{ij})$ for $X \in \mathbb{R}^{m \times n}$. The proximal operator of $\alpha\phi$ maps the matrix X to the matrix $\text{prox}_{\alpha\phi}(X) = A \in [0, 1]^{m \times n}$*

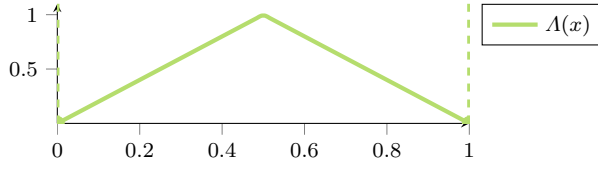


Fig. 4: The function Λ penalizing non-binary values.

defined by $A_{ji} = \text{prox}_{\alpha\Lambda}(X_{ji})$, where for $x \in \mathbb{R}$ it holds that

$$\text{prox}_{\alpha\Lambda}(x) = \begin{cases} \max\{0, x - 2\alpha\} & x \leq 0.5 \\ \min\{1, x + 2\alpha\} & x > 0.5. \end{cases} \quad (9)$$

This enables a minimization according to the cost measure of ASSO, setting $G(X, Y) = 0$. However, without a generalizing term it is unlikely that the rank is properly identified; the loss function certainly attains a minimum when one of the factor matrices is equal to the data matrix and the other one is the identity. Thus, we seek cost measures which are suitable for a minimization within PAL-TILING and whose application results in an algorithm which is capable to identify the correct rank.

3.3 PANPAL

The cost measure f_{L1} (as applied by PANDA) can easily be integrated into PAL-TILING. Since the proximal operator ensures that the factor matrices in all steps have values between zero and one, the $L1$ -norm of the factor matrices equates a simple summation over all matrix entries. Thus, the $L1$ -norm is a smooth function on the nonnegative domain of the factor matrices and can be used as regularizing function. We call the resulting algorithm PANPAL as it employs the cost measure of **Pan**DA in the minimization technique **PAL**-TILING:

PANPAL: Apply PAL-TILING with

– Cost measure

$$f_{L1}(X, Y, D) = |D - YX^T| + |X| + |Y|$$

– Relaxed objective

$$F(X, Y, D) = \frac{1}{2}\|D - YX^T\|^2 + \frac{1}{2}(|X| + |Y|)$$

– Partial Gradients

$$\nabla_X F(X, Y, D) = (YX^T - D)^T Y + (0.5)_{is}$$

$$\nabla_Y F(X, Y, D) = (YX^T - D)X + (0.5)_{js}$$

– Lipschitz moduli

$$M_{\nabla_X F}(Y) = \|YY^T\|$$

$$M_{\nabla_Y F}(X) = \|XX^T\|$$

3.4 PRIMP

So far, the cost measure of KRIMP has been disregarded in the context of Boolean matrix factorization. Since the traditionally employed cover function is incompatible with overlapping patterns or patterns which cover more items than persistent in the transaction, the task to find the best encoding by code tables is associated with the sub-domain of pattern mining (Miettinen and Vreeken 2014; Lucchese et al 2014; Karaev et al 2015). The definition of the long-established cover function is heuristically determined under the assumption that there is one globally valid cover function which is applicable on all datasets if a suitable code table is found. Although this approach might be favorable in sub-domains like classification or detection of changes in a data stream (Vreeken et al 2011; Van Leeuwen and Siebes 2008), it is current best practice in the domain of tiling to take negative noise into account (see Sec. 2.2). Thus, we break away from the conventional view on the cover function as a predefined instance and regard it as an extrapolation of the mapping from patterns to transactions which is defined by the matrix Y . Thereby, we intend to learn a suitable pair of code table and cover function for every dataset. This is motivated by the following observation.

Lemma 1 *Let D be a data matrix. For any code table CT and its cover function there exists a Boolean matrix factorization $D = \theta(YX^T) + N$ such that non-singleton patterns in CT are mirrored in X and the cover function is reflected by Y . The description lengths correspond to each other, such that*

$$L_{CT}(D, CT) = f_{CT}(X, Y, D) = f_{CT}^D(X, Y, D) + f_{CT}^M(X, Y, D),$$

where the functions returning the model and the data description size are given as

$$\begin{aligned} f_{CT}^D(X, Y, D) &= - \sum_{s=1}^r |Y_{\cdot s}| \cdot \log(p_s) - \sum_{i=1}^n |N_{\cdot i}| \cdot \log(p_{r+i}) \\ &= L_{CT}^D(D, CT) \\ f_{CT}^M(X, Y, D) &= \sum_{s: |Y_{\cdot s}| > 0} (X_{\cdot s}^T c - \log(p_s)) + \sum_{i: |N_{\cdot i}| > 0} (c_i - \log(p_{r+i})) \\ &= L_{CT}^M(CT). \end{aligned}$$

The probabilities p_s and p_{r+i} indicate the relational usage of non-singleton patterns $X_{\cdot s}$ and singletons $\{i\}$,

$$p_s = \frac{|Y_{\cdot s}|}{|Y| + |N|}, \quad p_{r+i} = \frac{|N_{\cdot i}|}{|Y| + |N|}.$$

We denote with $c \in \mathbb{R}_+^n$ the vector of standard code lengths for each item, i.e.,

$$c_i = - \log \left(\frac{|D_{\cdot i}|}{|D|} \right).$$

The proof of this lemma can be found in Appendix B. We remark that this formulation also puts new emphasis on the debate about the model definition in this MDL application. As commented by Siebes and Kersten (2011), the cover function actually is a part of the model and if we learn the cover function together

with the code table, an encoding of the data is not possible if only the code table is present. However, to be in line with common practice in the field we stick with the description length computation as originally proposed by Siebes et al (2006), which is also used in Lemma 1 and makes our results comparable to previously published results.

The transfer from a code table encoding to a Boolean matrix factorization provides another view on the objective of KRIMP-related algorithms. While the focus of matrix factorizations lies on the extraction of a given ground truth, the originally formulated task aims at the derivation of subjectively interesting patterns – equating interestingness with the ability to compress. Moreover, the tiling derived by Boolean matrix factorizations obliges certain requirements such as the linear independence of columns/rows and the bound on the rank ($r \leq \min\{m, n\}$) which follows from that.

Considering, in reverse, the transfer from a matrix factorization to an encoding by code tables, we naturally receive access to the treatment of negative noise. We can calculate the description size f_{CT} for arbitrary factor matrices, even if the resulting noise matrix contains negative entries. Yet, the question arises if this also has a suitable interpretation with regard to the encoding. In fact, the interpretation is simple: the items having a negative noise entry can be transmitted just as the items with positive noise entries; their singleton codes are appended to the belonging transaction. If the item is not covered by any other pattern used in this transaction, then it belongs to a positive noise entry and otherwise to a negative one. We obtain therewith a description length equal to f_{CT} . That is, each transaction D_j is described by the code concatenation of patterns $X_{\cdot r}$ where $Y_{jr} = 1$ and the singleton codes of items i having $N_{ji} \neq 0$.

However, the compression size $f_{CT}(X, Y, D)$ is not continuous. There are points of discontinuity at encodings which do not use a pattern present in X or one of the singletons. To approximate this function as required in PAL-TILING (Eq. (5)), we assume that each pattern in X is used at least once. For singletons, we do not wish to make such an assumption; usages of singletons are reflected in the noise matrix and we want to keep the noise as small as possible. Therefore, we bound the description size which is induced by singletons by the RSS. Then, we obtain a smooth function which meets the requirements of PAL-TILING. This is specified by the following theorem whose proof can be found in Appendix C.

Theorem 2 *Given binary matrices X and Y and $\mu = 1 + \log(n)$, it holds that*

$$f_{CT}^D(X, Y, D) \leq \mu \|D - YX^T\|^2 - \sum_{s=1}^r (|Y_{\cdot s}| + 1) \log \left(\frac{|Y_{\cdot s}| + 1}{|Y| + r} \right) + |Y| \quad (10)$$

So far, this bound encompasses the description size of the data, yet the description size of the model is also discontinuous at points where one of the patterns is not used at all. The description size of one side of the code table, representing the patterns by standard singleton codes c , computed by

$$|X^T c| = \sum_{s=1}^r X_{\cdot s}^T c \geq \sum_{s: |Y_{\cdot s}| > 0} X_{\cdot s}^T c,$$

can easily be integrated into the smooth approximation. The matrix X and the standard code sizes c_i contain only nonnegative entries, thus the computation of

the $L1$ -norm boils down to a summation over all entries in the vector $X^T c$. The remaining terms which compose the model complexity are bounded above by a constant, due to the fixation of the rank during the minimization of the relaxed objective. Thus, we minimize the relaxed function as denoted in the box below. The required Lipschitz constants are computed in Appendix D. We refer to this algorithm as PRIMP, as it performs **PAL-TILING** with the objective of **Krimp**.

PRIMP: Apply PAL-TILING with

– Constants

$$c = (c_i)_{i \in \mathcal{I}}, \quad c_i = -\log \left(\frac{|D_{\cdot i}|}{|D|} \right)$$

$$\mu = 1 + \log(n)$$

– Cost measure

$$f_{\text{CT}}(X, Y, D)$$

– Relaxed objective

$$F(X, Y, D) = \frac{\mu}{2} \|D - YX^T\|^2 + \frac{1}{2} G(X, Y)$$

$$G(X, Y) = -\sum_{s=1}^r (|Y_{\cdot s}| + 1) \log \left(\frac{|Y_{\cdot s}| + 1}{|Y| + r} \right) + |X^T c| + |Y|$$

– Partial Gradients

$$\nabla_X F(X, Y, D) = \mu(YX^T - D)^T Y + c(0.5)_s^T$$

$$\nabla_Y F(X, Y, D) = \mu(YX^T - D)X - \frac{1}{2} \left(\log \left(\frac{|Y_{\cdot s}| + 1}{|Y| + r} \right) \right)_{j_s} + (0.5)_{j_s}$$

– Lipschitz moduli

$$M_{\nabla_X F}(Y) = \mu \|YY^T\|$$

$$M_{\nabla_Y F}(X) = \mu \|XX^T\| + m$$

4 Experiments

We conduct experiments on a series of synthetic data matrices, exploring the ability to detect the planted tiling, i.e., to recover generated matrices X and Y in presence of various noise structures. In real-world data experiments we compare the costs of obtained models in various measures. Also, we perform a qualitative evaluation of the factor methods, visualizing the algorithms' understanding of tiles and noise on the basis of images. We compare the PAL-TILING instances PANPAL and PRIMP with the available implementations of PANDA+², MDL4BMF³

² <http://hpc.isti.cnr.it/~claudio/web/archives/20131113/index.html>

³ <http://people.mpi-inf.mpg.de/~skaraev/>

and NASSAU³. Concerning PANPAL and PRIMP, we apply $K = 50,000$ iterations and try thresholds with step size 0.05, i.e., $T = \{0.05k \mid k \in \{0, 1, \dots, 20\}\}$. We apply the same set of thresholds T to MDL4BMF, which is also the average increment used in experiments by Miettinen and Vreeken (2014); Lucchese et al (2014); Karaev et al (2015). For PANDA+ we choose the TypedXOR measure and use 20 randomization rounds and correlating items as suggested in the literature Lucchese et al (2014). Apart from that, the default settings apply.

We exclude SLIM from our experiments as it can not be fairly compared to Boolean matrix factorization algorithms. To illustrate, SLIM returns far more patterns (500 to 3000 monotonically increasing with noise) than planted (25) in our synthetic data sets. Hence, a depiction of this algorithm’s rank would distort the rank charts due to its unreasonable performance.

For synthetic and real world experiments, we set the rank increment $\Delta_r = 10$; sensitivity to this parameter is explored in Sec. 4.5. For our image evaluation, we set (if possible) a maximum number of 10 returned tiles and depict the four most informative tiles. Here, we set $\Delta_r = 1$, to consistently allow for more factorization rounds in case of higher estimated rank.

A separate run time comparison of the aforementioned algorithms is not conducted. This is because we can not guarantee that the underlying data structures and platform specific optimizations are equally well tuned, especially for the approaches for which we make use of the reference implementation (PANDA+, MDL4BMF and NASSAU). Note, however, that due to the formulation of PAL-TILING in terms of linear algebra, a highly parallel implementation on graphics processing units (GPU) is straightforward. Therefore, experiments regarding PANPAL and PRIMP are executed on a GPU with 2688 arithmetic cores and 6GiB GDDR5 memory. The run time of the GPU based algorithms is about 50 times lower, compared to the ordinary implementations, e.g., a task that is finished by PRIMP in a few seconds, requires 30 minutes by MDL4BMF. We provide the source code of our algorithms together with the data generating script⁴.

4.1 Synthetic Data Generation

We generate data matrices according to the scheme established by Miettinen and Vreeken (2014); Karaev et al (2015) and Lucchese et al (2014). Yet, we constrain the set of generated factor matrices to contain at least one percent of uniquely assigned ones to ensure linear independence of column vectors. This ensures that for $r^* \leq n, m$ and generated matrices $X^* \in \{0, 1\}^{n \times r^*}$ and $Y^* \in \{0, 1\}^{m \times r^*}$, the matrix $D = Y^* X^{*T}$ indeed has rank r^* . We describe the data generation process as a function from dimensions n and m , rank r^* , density parameter q and noise probabilities p_+ and p_- .

GenerateData (n, m, r^*, q, p_+, p_-)

⁴ <http://sfb876.tu-dortmund.de/primp>

Variation	p_+ [%]	p_- [%]	r	q	Density [%]	Overlap [%]
Uniform Noise	0	0	25	0.1	6.6 ± 0.8	2.3 ± 0.3
	25	25	25	0.1	28.3 ± 0.4	2.3 ± 0.3
Pos/Neg Noise	25	3	25	0.1	30.6 ± 0.4	3.0 ± 0.4
	3	25	25	0.1	8.5 ± 0.4	2.9 ± 0.5
Rank	10	10	5	0.1	11.3 ± 0.4	0.3 ± 0.2
	10	10	45	0.1	18.6 ± 0.9	8.7 ± 1.0
Density	10	10	25	0.1	15.3 ± 0.6	2.3 ± 0.3
	10	10	25	0.3	40.6 ± 4.3	26.9 ± 6.1

Table 2: Characteristics of generated datasets. The values are aggregated over eight generated datasets, four for each combination of dimensions $(n, m) \in \{(500, 1600), (800, 1000)\}$. Overlap denotes the percentage of overlapping entries in relation to the region covered by all tiles together and density is the region covered by all tiles together in relation to nm .

1. Set $k = \lceil \frac{n}{100} \rceil$ and $l = \lceil \frac{m}{100} \rceil$. Let $\mathbf{1}_k$ and $\mathbf{1}_l$ denote the k - and l -dimensional vector filled with ones. Draw factor matrices of the form

$$X^* = \begin{pmatrix} \mathbf{1}_k & & 0 \\ & \ddots & \\ 0 & & \mathbf{1}_k \\ | & & | \\ x_1 & \cdots & x_{r^*} \\ | & & | \end{pmatrix}, \quad Y^* = \begin{pmatrix} \mathbf{1}_l & & 0 \\ & \ddots & \\ 0 & & \mathbf{1}_l \\ | & & | \\ y_1 & \cdots & y_{r^*} \\ | & & | \end{pmatrix},$$

where we draw for $1 \leq s \leq r^*$, $\tilde{n} = n - kr^*$ and $\tilde{m} = m - lr^*$

- $x_s \in \{x \in \{0, 1\}^{\tilde{n}} \mid |x| \leq \lfloor q\tilde{n} \rfloor\}$ uniformly random
 - $y_s \in \{y \in \{0, 1\}^{\tilde{m}} \mid |y| \leq \lfloor q\tilde{m} \rfloor\}$ uniformly random
2. Set $D = Y^*X^{*T} + N$ with noise matrix N generated by the following scheme
 - If $D_{ji} = 0$ then $N_{ji} = 1$ with probability p_+
 - If $D_{ji} = 1$ then $N_{ji} = -1$ with probability p_-

We generate datasets for distinct settings with dimensions $(n, m) \in \{(500, 1600), (1600, 500), (800, 1000), (1000, 800)\}$, $r \in [5, 25]$, $q \in [0.1, 0.3]$ and $p_{\pm} \in [0, 25]$. Table 2 summarizes the basic statistics of the generated datasets.

4.2 Measuring the Tiling Quality

We quantify how well a computed tiling (X, Y) matches the planted tiling (X^*, Y^*) by an adaptation of the micro-averaged F-measure, known from multi-class classification tasks. In this regard, we identify a planted tile $Y_{\cdot s}^* X_{\cdot s}^{*T}$ with a class which contains the tuples (j, i) which indicate ones. Then, a suitable one-to-one matching σ between computed and planted tiles allows to compare the *true* labels $Y_{j s}^* X_{i s}^{*T}$ with the *predicted* labels $Y_{j \sigma(s)} X_{i \sigma(s)}^T$. Therewith, we can naturally calculate precision and recall and finally the F -measure.

We assume w.l.o.g. that $X, X^* \in \{0, 1\}^{n \times r}$ and $Y, Y^* \in \{0, 1\}^{n \times r}$, otherwise we attach zero columns to the matrices such that the dimensions match. We compute with the Hungarian algorithm (Kuhn 1955) a permutation $\sigma : \{1, \dots, r\} \rightarrow \{1, \dots, r\}$ which matches computed and planted tiles one-to-one such that $\sum_{s=1}^r F_{s, \sigma(s)}$ is maximized. The $F_{s,t}$ -measure is calculated for $1 \leq s, t \leq r$ by

$$F_{s,t} = 2 \frac{\text{pre}_{s,t} \cdot \text{rec}_{s,t}}{\text{pre}_{s,t} + \text{rec}_{s,t}},$$

where $\text{pre}_{s,t}$ and $\text{rec}_{s,t}$ denote precision and recall between the planted tile $(X_{\cdot,s}^*, Y_{\cdot,s}^*)$ and computed tile $(X_{\cdot,t}, Y_{\cdot,t})$

$$\text{pre}_{s,t} = \frac{|(Y_{\cdot,s}^* \circ Y_{\cdot,t})(X_{\cdot,s}^* \circ X_{\cdot,t})^T|}{|Y_{\cdot,t} X_{\cdot,t}^T|} \quad \text{rec}_{s,t} = \frac{|(Y_{\cdot,s}^* \circ Y_{\cdot,t})(X_{\cdot,s}^* \circ X_{\cdot,t})^T|}{|Y_{\cdot,s}^* X_{\cdot,s}^{*T}|}.$$

Having computed the perfect matching σ , we calculate precision and recall for the obtained factorization by

$$\begin{aligned} \text{pre} &= \frac{\sum_{s=1}^r |(Y_{\cdot,s}^* \circ Y_{\cdot,\sigma(s)})(X_{\cdot,s}^* \circ X_{\cdot,\sigma(s)})^T|}{\sum_{s=1}^r |Y_{\cdot,s} X_{\cdot,s}^T|} = \frac{|(Y^* \circ Y_{\cdot,\sigma(\cdot)})(X^* \circ X_{\cdot,\sigma(\cdot)})^T|}{|Y X^T|} \\ \text{rec} &= \frac{\sum_{s=1}^r |(Y_{\cdot,s}^* \circ Y_{\cdot,\sigma(s)})(X_{\cdot,s}^* \circ X_{\cdot,\sigma(s)})^T|}{\sum_{s=1}^r |Y_{\cdot,s}^* X_{\cdot,s}^{*T}|} = \frac{|(Y^* \circ Y_{\cdot,\sigma(\cdot)})(X^* \circ X_{\cdot,\sigma(\cdot)})^T|}{|Y^* X^{*T}|}. \end{aligned}$$

The micro F -measure is defined in terms of precision and recall as defined above. This is equivalent to a convex combination of the $F_{s, \sigma(s)}$ -measurements:

$$F = 2 \frac{\text{pre} \cdot \text{rec}}{\text{pre} + \text{rec}} = \sum_{s=1}^r \frac{|Y_{\cdot,s}^* X_{\cdot,s}^{*T}| + |Y_{\cdot,\sigma(s)} X_{\cdot,\sigma(s)}^T|}{|Y^* X^{*T}| + |Y X^T|} F_{s, \sigma(s)}.$$

The F -measure has values between zero and one. The closer it approaches one, the more accurate the obtained tiling is. The plots which display the F -measure indicate the average value with error bars having the length of twice the standard deviation.

We express the values of involved cost measures in relation to the empty model

$$\%f(X, Y, D) = \frac{f(X, Y, D)}{f(\mathbf{0}_n, \mathbf{0}_m, D)} \cdot 100.$$

4.3 Make some Noise

In the following series of experiments, varying the noise, we plot the F -measure and the rank of the returned tiling against the percentage of noise which is added. The planted factorization has a rank of $r^* = 25$ and density parameter $q = 0.1$. The noise level varies from 0% to 25% as displayed on the x -axis.

First, we compare the effects of the matrix dimensions and aggregate results over 10 generated matrices with dimensions 800×1000 and 500×1600 together with their transpose, as depicted in Figs. 5 and 6. Comparing the results for a data matrix and its transpose is particularly interesting for the algorithm PRIMP. Since it applies different regularizations on X and Y , we want to assess how this affects

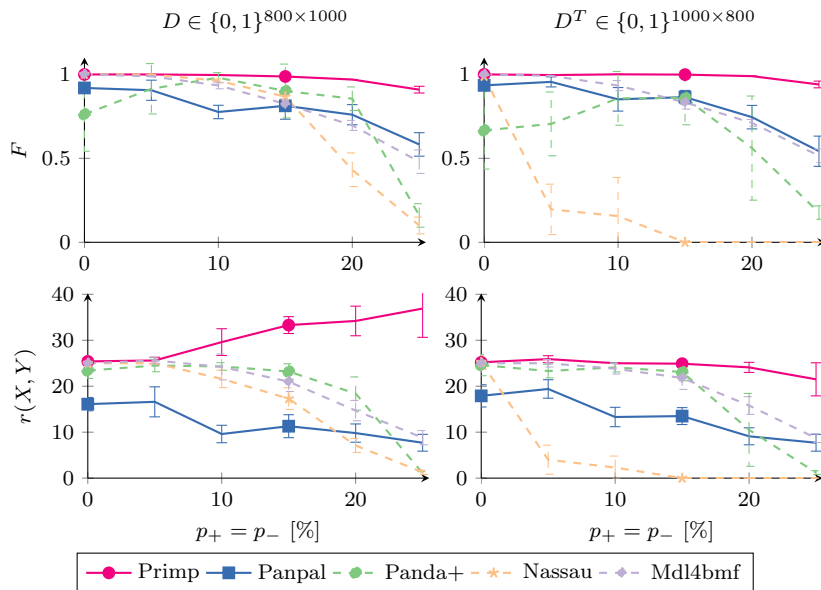


Fig. 5: Variation of uniform noise for 800×1000 and 1000×800 dimensional data. Comparison of F -measures (the higher the better) and the estimated rank of the calculated tiling (the closer to 25 the better) for varying levels of noise, i.e., $p_+ = p_-$ is indicated on the x-axis (best viewed in color).

the results of PRIMP in practice. The remaining algorithms minimize an objective which is invariant to a transposition of the input matrix. It is desirable that this is also reflected in practice.

We observe from Figs. 5 and 6 that the algorithms likely return fewer tiles the more the noise increases. This culminates in the replication of almost none of the tiles at highest noise level for the algorithms PANDA+ and NASSAU. NASSAU particularly strongly underestimates the rank if the data matrix is transposed, i.e., $n > m$. In this case, NASSAU returns close or equal to zero tiles, even if the noise is low. PANDA+ yields correct rank estimations up to a noise of 15%, but its fluctuating F -measure indicates that planted tiles are not correctly recovered after all. In particular, its F -values differ from the untransposed to the transposed case even if the rank estimations are similar and close to r^* . MDL4BMF shows a robust behavior towards a transposition of the the input matrix. Its suitable rank estimations up to a noise of 15% are mirrored in a high F -measure. PANPAL consistently underestimates the rank, yet can achieve comparatively high F -measures. Its results exhibit minor deviations from the untransposed to the transposed case. Recognizable differences occur when n and m differ more widely (Fig. 6) and the noise level is low. Under these circumstances, PANPAL yields higher rank estimations if the matrix is transposed. We note, that the code of PAL-TILING and therewith also the code of PANPAL inhibits only one distinction between X and Y , which is the order in which gradient steps are invoked. Whether this actually influences the output of the algorithm is an interesting question but it is beyond the scope

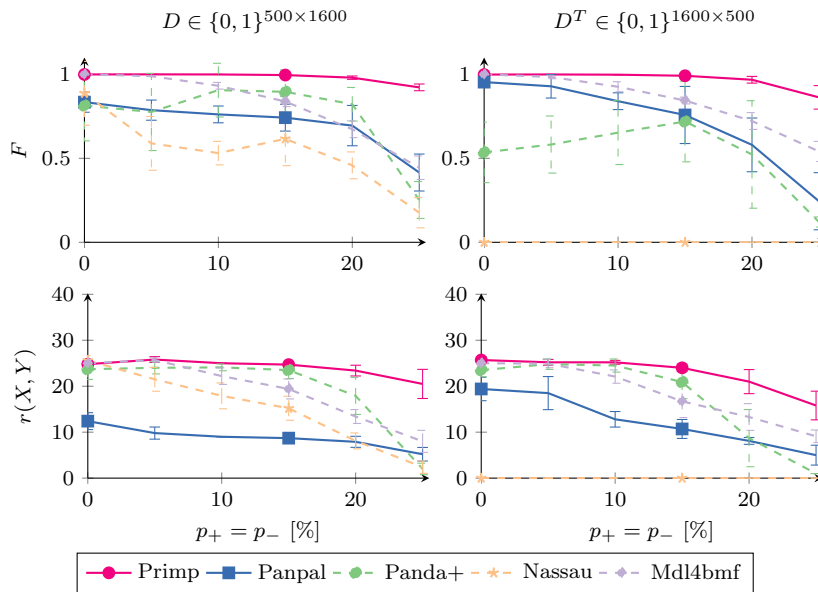


Fig. 6: Variation of uniform noise for 500×1600 and 1600×500 dimensional data. Comparison of F -measures (the higher the better) and the estimated rank of the calculated tiling (the closer to 25 the better) for varying levels of noise, i.e., $p_+ = p_-$ is indicated on the x-axis (best viewed in color).

of this paper. PRIMP is characterized by overall high values in the F -measure. It has a tendency to estimate the rank higher in the untransposed case, i.e., if $m > n$. This is particularly notably if the matrices are almost square (Fig. 5). This suggests that the cost measure favors modeling tiles having fewer items and more transactions. That aside, the overall high F -measure shows that additionally modeled tiles cover only a small area in comparison to planted ones.

In Fig. 7 we contrast varying distributions of positive and negative noise (p_+ and p_-). From here on, we aggregate results over eight matrices, two for each of the considered matrix dimensions. However, we make an exception for NASSAU and transpose the input matrix if $n > m$, as NASSAU tends to return zero tiles in this case.

On the left of Fig. 7, we show the aggregated results when varying uniform noise, as discussed for individual dimensions before. All algorithms except for PRIMP tend to return fewer tiles with increasing noise. Despite correct rank estimations, PANDA+ displays volatile F -measure values. PRIMP's rank estimations are correct in the mean, but variance is quite high.

The middle plot depicts variations of negative noise while positive noise is fixed to 3%. In this setting, the algorithms PRIMP, MDL4BMF and NASSAU are capable of identifying the planted tiling for all noise levels. The suitability of NASSAU in the prevalence of negative noise corresponds to the experimental evaluation by Karaev et al (2015). MDL4BMF and PRIMP yield equally appropriate results in this experiment. The approximations of PANDA+ and PANPAL are notably less

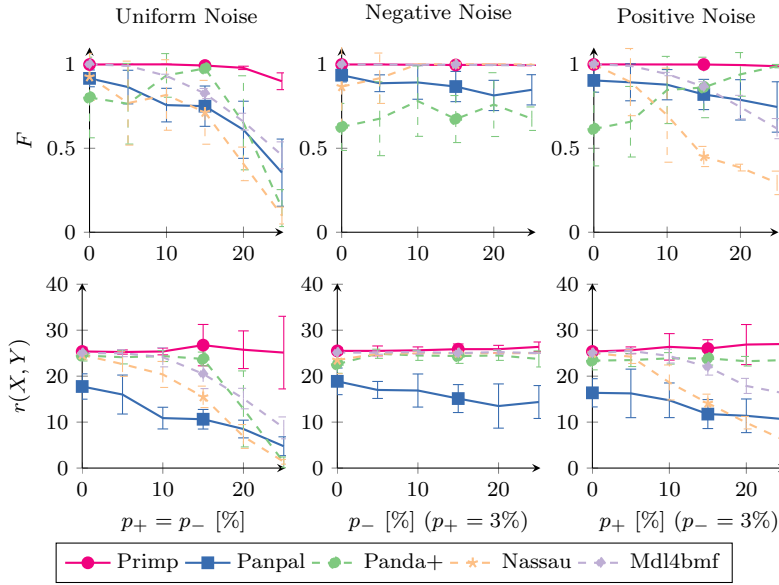


Fig. 7: Variation of uniform, positive and negative noise. Comparison of F -measures (the higher the better) and the estimated rank of the calculated tiling (the closer to 25 the better) for varying levels of noise, i.e., p_+ and p_- are indicated on the x-axis (best viewed in color).

accurate. Although PANDA+ correctly estimates the rank around 25 and PANPAL's estimations lie between 10 and 20, PANPAL achieves higher F -measures than PANDA+.

The plots on the right of Fig. 7 show the impact of variations on the positive noise, fixing the negative noise to 3%. Here, NASSAU, MDL4BMF and PANPAL tend to underestimate the rank the more the noise increases, similarly to but not as drastic as in experiments with uniformly distributed noise. PANDA+ shows a poor recovery of planted coherent tiles at 0% positive noise, but its F -value peculiarly increases with increasing positive noise. PRIMP robustly identifies the true tiling for all levels of noise, yet inhibits a higher variance from the mean of the rank estimations.

4.4 Variation of Tiling Generation Parameters

We present effects on variations from the rank in Fig. 8 whereby the default parameters of 10% uniform noise and $q = 0.1$ apply. We observe a hierarchy of algorithms in the tendency to underestimate the rank throughout all values of r^* . By far the lowest rank estimations are returned by PANPAL, followed by NASSAU, MDL4BMF, PANDA+ and PRIMP. PANDA+ and PRIMP consistently return accurate rank estimations. It is remarkable that for ranks higher than 30, PANPAL obtains

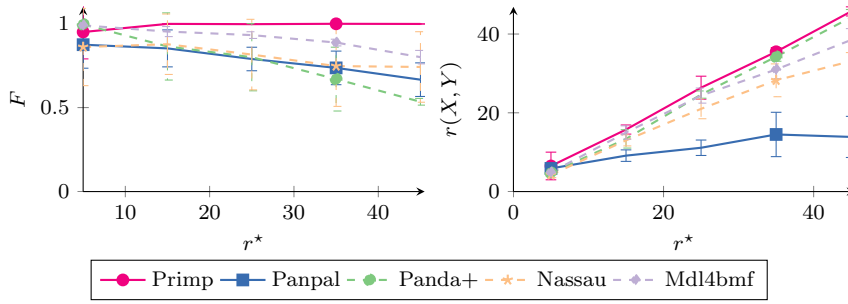


Fig. 8: Variation of the rank $r^* \in \{5, \dots, 45\}$ of the planted tiling. Comparison of F -measures (the higher the better) and estimated rank (the closer to the identity function the better) of calculated tilings for uniform noise of $p_+ = p_- = 10\%$ (best viewed in color).

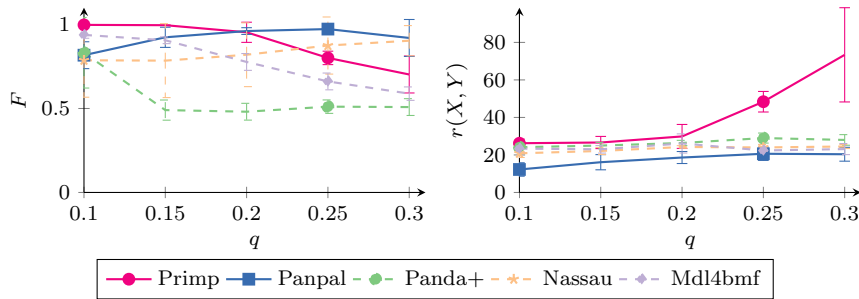


Fig. 9: Variation of density and overlap influencing parameter $q \in [0.1, \dots, 0.3]$. Comparison of F -measures (the higher the better) and the estimated rank of the calculated tiling (the closer to 25 the better) for uniform noise of $p_+ = p_- = 10\%$ (best viewed in color).

higher F values than PANDA+ despite of modeling only a fraction of the planted tiles. PRIMP provides a steadily accurate recovery of planted tiles.

In Fig. 9 we vary the density and overlap influencing parameter q , which determines the maximum density of a column vector in X and Y . We observe two classes of algorithms. The first class, consisting of PRIMP, PANDA+ and MDL4BMF decreases in the F -measure with increasing q . In this class, PRIMP always retrieves highest F -values. In return, the F -values from the second class of PANPAL and NASSAU increase with q . Here, PANPAL bounds the F -values of NASSAU from above. Correspondingly, PRIMP and PANPAL have a *break-even-point* at $q = 0.2$. From this value on, PRIMP starts to considerably overestimate the rank while PANPAL's tendency to underestimate the rank, decreases. For $q \geq 0.2$, PANPAL estimates the rank close to 20 in average. That is, five planted tiles are not modeled in average. Still, the F -measure indicates that for the denser and more overlapping datasets, PANPAL most accurately discovers the planted tiles.

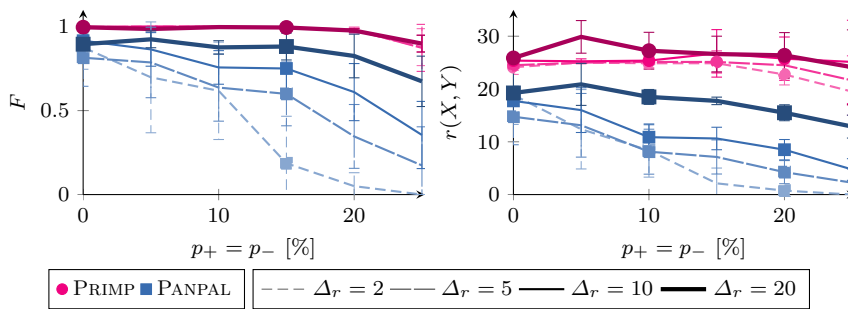


Fig. 10: Variation of rank increment $\Delta_r \in \{2, 5, 10, 20\}$. Comparison of F -measures (the higher the better) and the estimated rank of the calculated tiling (the closer to 25 the better) for uniform noise of $p_+ = p_-$ indicated by the x -axis (best viewed in color).

4.5 Sensitivity to the Rank Increment

In the default setting of our synthetic experiments, the PAL-TILING algorithms PRIMP and PANPAL have to increase the rank two times by $\Delta_r = 10$ to estimate the rank $r^* = 25$ correctly. In the experiments varying the rank, we have seen that PRIMP is able to find the correct rank if twice as many decisions correctly have to be made. Here, we want to assess how robust the performance of PAL-TILING algorithms to the parameter Δ_r is. What happens if, e.g., $\Delta_r = 2$ and 23 rank increments have to be administered correctly?

Fig. 10 shows F -measurements and estimated ranks of the algorithms PRIMP and PANPAL, invoked with diverse rank increments $\Delta_r \in \{2, 5, 10, 20\}$ on datasets with varying uniform noise. It is noticeable that the rank estimations of PANPAL rapidly diverge with increasing noise while the plots of PRIMP stay comparatively close. PANPAL's tendency to underestimate the rank grows for smaller rank increments. In return, the rank estimations of PANPAL can be improved by choosing a large rank increment, i.e. $\Delta_r \approx r^*$. However, since we do not know the rank in real world applications, different increment values have to be tried and compared, contradicting our goal to automatically determine this parameter. Still, PANPAL yields potentially useful lower bounds on the actual rank.

The average rank estimations of PRIMP have a maximum aberration of five from the actual rank throughout all noise variations. The graphical display of $r(X, Y)$ for $\Delta_r = 20$ has a peak at 5% uniform noise but is close to r^* otherwise. For rank increments smaller than 10, the estimations do not distinctly decrease until the noise exceeds 20%. Here, a rank increment of $\Delta_r = 5$ yields the most accurate rank estimations, having also lowest standard deviations from the mean. Particularly, PRIMP's tendency to overestimate the rank in specific settings can be corrected by choosing smaller rank increments. Nonetheless, all these rank deviations barely effect the F -measure, which demonstrates the robustly well fitted recovery of the underlying model regardless of the choice of rank increment.

	Algorithm	\bar{F}	$\overline{\%f_{RSS}}$	$\overline{\%f_{CT}}$	$\overline{\%f_{L1}}$	$\overline{\%f_{TXD}}$
$p_{\pm} = 25\%$	Planted	1.0 \pm 0.0	88.37 \pm 1.24	89.88 \pm 1.25	89.51 \pm 1.25	96.5 \pm 0.61
	PRIMP	0.9 \pm 0.05	89.58 \pm 1.71	91.0 \pm 1.54	90.65 \pm 1.59	97.0 \pm 0.62
	PANPAL	0.35 \pm 0.2	97.17 \pm 1.62	97.56 \pm 1.46	97.47 \pm 1.49	99.16 \pm 0.56
	MDL4BMF	0.46 \pm 0.08	96.6 \pm 0.86	97.25 \pm 0.76	97.11 \pm 0.77	99.2 \pm 0.25
	PANDA	0.14 \pm 0.11	99.14 \pm 0.77	99.28 \pm 0.62	99.24 \pm 0.66	99.75 \pm 0.19
	NASSAU	0.1 \pm 0.05	100.5 \pm 0.29	100.7 \pm 0.3	100.69 \pm 0.31	99.75 \pm 0.15
$r^* = 45$	Planted	1.0 \pm 0.0	50.27 \pm 1.25	54.67 \pm 1.29	53.29 \pm 1.29	69.77 \pm 0.96
	PRIMP	1.0 \pm 0.0	50.32 \pm 1.23	54.74 \pm 1.27	53.35 \pm 1.27	69.85 \pm 0.93
	PANPAL	0.67 \pm 0.1	73.0 \pm 6.04	75.04 \pm 5.56	74.29 \pm 5.71	84.66 \pm 3.78
	MDL4BMF	0.8 \pm 0.04	62.67 \pm 1.41	66.72 \pm 1.53	65.48 \pm 1.46	79.21 \pm 1.03
	PANDA	0.53 \pm 0.02	89.02 \pm 1.76	92.34 \pm 2.16	92.76 \pm 2.09	86.0 \pm 0.7
	NASSAU	0.74 \pm 0.21	64.43 \pm 10.08	68.27 \pm 10.24	67.28 \pm 10.43	77.47 \pm 4.87
$q = 0.3$	Planted	1.0 \pm 0.0	24.94 \pm 2.74	27.84 \pm 2.78	27.11 \pm 2.7	51.97 \pm 1.04
	PRIMP	0.7 \pm 0.1	27.04 \pm 2.46	31.23 \pm 2.33	30.33 \pm 2.25	57.52 \pm 2.03
	PANPAL	0.92 \pm 0.11	29.45 \pm 3.17	31.98 \pm 3.06	31.31 \pm 3.1	57.09 \pm 3.86
	MDL4BMF	0.59 \pm 0.04	45.08 \pm 2.14	48.53 \pm 2.01	47.88 \pm 1.96	73.81 \pm 1.49
	PANDA	0.51 \pm 0.05	54.12 \pm 8.9	57.07 \pm 8.83	56.74 \pm 8.86	75.88 \pm 2.32
	NASSAU	0.9 \pm 0.09	29.11 \pm 5.19	32.07 \pm 5.2	31.42 \pm 5.2	56.79 \pm 4.2

Table 3: Average cost measures of computed and planted models, denoted relation to the costs of the empty model. For each setting (variation of one data generation parameter while the others are set to default values $r^* = 25$, $q = 0.3$ and $p_{\pm} = 25\%$) the average value is computed over all considered dimension variations.

4.6 Comparison of Cost Measures

We have seen how well the competing algorithms perform with regard to the F -measure. Then again, assessing the performance on real data requires other measurements. Possible candidates are the costs listed in Table 1. Subsequently, we relate selected costs of computed and planted models to the F -measure and discuss whether we can deduce a suitable extraction of the underlying model from a low cost measure; is smaller always better?

Table 3 displays the average costs in relation to the empty model for the four measures f_{RSS} , the residual sum of squares, f_{CT} , the compression size obtained by code tables, f_{L1} , the $L1$ -regularized residual sum of squares and f_{TXD} , the Typed XOR DtM measure. We examine three parameter settings, one for the highest value in each variation of the data generation parameters r^* , q and p_{\pm} . Thereby, default settings of $r^* = 25$, $q = 0.1$, $p_{\pm} = 10\%$ apply, if not stated otherwise. The values of the planted model are shaded out while the highest F -measure and lowest mean costs of computed models are highlighted.

We can trace that high F -values often correspond to lower costs, regardless of the measurement. This effect is immediately perceivable at rows where PRIMP attains highest F -values and all of its cost values are highlighted as well. Yet, the experiments for $q = 0.3$ display a more diverse ranking among the measurements. In this setting, PRIMP decidedly overestimates the rank but still obtains lowest costs in all but the f_{TXD} measure. PANPAL attains the highest F -value, closely followed by NASSAU. Both algorithms reach second or third lowest costs in f_{RSS} , f_{CT} and f_{L1} . The f_{TXD} costs reflect the order of F -values more suitably, NASSAU obtains lowest costs, closely followed by PANPAL and PRIMP. In brief, the costs of PRIMP, PANPAL and NASSAU are always close while only MDL4BMF and PANDA+

Dataset D	m	n	Density [%]
Abstracts	859	4977	1.02
Mushroom	8124	120	19.33
MovieLens5M	29980	9044	1.81
MovieLens500K	3329	3015	4.99
Chess	3196	75	49.33

Table 4: Characteristics of considered datasets: Number of rows m , number of columns n and density $|D|/(nm)$ in percent.

lie notably behind. Here, the deciding clue is given by the rank, which separates the close cost measurements of PRIMP, PANPAL and NASSAU by showing that slight improvements in the costs by PRIMP are achieved by a disproportionate increase of the rank.

While for $q = 0.3$, the f_{TXD} costs appear suitable to reflect an appropriate extraction of tiles, in the setting of $p_{\pm} = 25\%$ we observe another facet. Here, we see that NASSAU reaches the same average f_{TXD} costs as PANDA+ although NASSAU increases the RSS in comparison to the empty model. This is indicated by relative costs larger than 100% in all measurements but f_{TXD} . Still, the ranking of f_{TXD} costs matches the F -measure ranking but this example shows, that a compression with respect to the f_{TXD} description length can be achieved without adaptation to the data.

4.7 Real-World Data Experiments

We conduct experiments on five datasets, whose characteristics are summarized in Table 4. *Chess* and *Mushroom* are discretized benchmark UCI datasets having a comparatively high density and around 50 times more rows than columns. The *Abstracts* dataset indicates the presence of stemmed words, excluding stop-words, in all ICDM paper abstracts until 2007 (De Bie 2011). It is a sparse dataset with around 5 times as many columns (words) as rows (documents). Finally, the *MovieLens5M* and *MovieLens500K* are binarized versions of the *MovieLens10M*⁵ and *MovieLens1M*⁶ datasets, where rows correspond to users and columns to movies. We set $D_{ji} = 1$ iff user j recommends movie i with more than three out of five stars. After selecting only those users which recommend more than 50 movies and those movies which receive more than five recommendations, we obtain two datasets with a balanced number of rows and columns. The *MovieLens5M* dataset, containing 5M ones, and the *MovieLens500K* dataset, with 500 thousand ones, have a (as one would expect, due to the dataset domain) high amount of negative noise due to missing values. Originally, we intended to consider only the *MovieLens5M* dataset, but NASSAU and MDL4BMF could not terminate in reasonable time – we aborted the calculations after one month. Therefore, we also prepared the smaller *MovieLens500K* dataset. (For comparison: While PRIMP, PANPAL and PANDA+ require around ten minutes to compute the result for *MovieLens500K*, MDL4BMF

⁵ <http://grouplens.org/datasets/movielens/10m/>

⁶ <http://grouplens.org/datasets/movielens/1m/>

Data	Algorithm	Rank	$\%f_{RSS}$	$\%f_{CT}$	$\%f_{L1}$	$\%f_{TXD}$		
Abstracts	PRIMP	46	93.0	98.6	96.33	96.12		
	PANPAL	1	99.8	99.96	99.89	99.74		
	MDL4BMF	24	95.84	100.68	100.49	97.05		
	NASSAU	3	99.81	101.76	103.05	96.84		
	PANDA+	133	113.34	125.27	140.49	88.19		
Chess	PRIMP	18	24.61	31.3	29.32	62.8		
	PANPAL	6	40.76	46.71	45.67	78.92		
	MDL4BMF	3	35.91	39.34	39.51	68.88		
	NASSAU	10	31.92	39.03	38.94	65.78		
	PANDA+	27	25.76	36.58	35.74	65.01		
MovieLens	500K	PRIMP	78	88.59	93.29	91.4	89.37	
		PANPAL	15	94.05	95.92	94.87	92.26	
		MDL4BMF	56	89.65	94.97	93.43	88.72	
		NASSAU	29	111.15	118.47	120.58	85.89	
		PANDA+	120	160.93	165.61	168.58	79.49	
	5M	PRIMP	209	89.31	93.14	91.2	88.16	
		PANPAL	38	93.68	95.72	94.39	88.73	
		PANDA+	1919	181.42	202.87	201.45	72.23	
		Mushroom	PRIMP	14	35.75	40.89	40.25	56.09
			PANPAL	7	44.03	51.23	48.52	63.75
MDL4BMF	87		23.39	36.6	32.47	50.37		
NASSAU	65		40.60	58.77	54.93	50.62		
PANDA+	40		100.30	117.34	112.80	66.98		

Table 5: Comparison of cost measures for real-world datasets.

MovieLens	PRIMP	PANPAL	MDL4BMF	NASSAU	PANDA+
500K	2.38	2.23	3.68	10.33	18.78
5M	2.08	2.78	-	-	23.14

Table 6: Percentage of traceable wrong recommendations of computed models for the MovieLens datasets, i.e., the relative amount of user-movie recommendations which correspond to bad reviews (< 2.5 stars out of five).

and NASSAU need more than five days.) Furthermore, we note that we transpose the *Abstracts* dataset for NASSAU, as it returns zero tiles otherwise.

We state the estimated rank and the attained costs, relative to the costs of the empty model for every considered dataset and algorithm in Table 5. The lowest costs are highlighted for each measure and dataset. We observe, similarly to the evaluation in Sec. 4.2, a tendency toward compliance among all measures, except for the Typed XOR description length. As such, PRIMP mostly obtains minimal costs in all datasets but *Mushroom*, where MDL4BMF reaches lowest costs. The models of PANDA+ exhibit for sparse datasets low Typed XOR DtM costs although the fit to the data is low ($f_{RSS} > 100\%$). The discrepancy between the f_{TXD} compression size and the other measurements is most remarkably for the *MovieLens* datasets. Here, the ranking with respect to f_{TXD} is almost inverse to the ranking with respect to other costs.

This leads to the question which cost measure indicates the suitable tiling in such situations? Luckily, we have for the MovieLens data the possibility to assess

how many recommendations would fail by the submitted bad reviews, which are not reflected in the input data. We state the relative amount of recommendations which correspond to bad reviews, i.e., $|D_{-\circ\theta}(YX^T)|/|D_{-}|$ where D_{-} is the matrix having $D_{ji} = 1$ iff user j rates movie i with less than 2.5 of five stars, in Table 6. We observe that the lower f_{TXD} costs are, the higher is the rate of recommendation failures, regardless of the estimated rank. Therefore, we expect PRIMP to discover the most liable grouping of users and movies, having the lowest approximation error and a very low ration of traceable wrong recommendations. Similarly, it is questionable if low f_{TXD} costs indicate suitable models in specific cases where the approximation error diverges such as for the *Abstracts* dataset.

4.8 Qualitative Inspection of Mined Tiles

The F -measure gives a hint at the kind of tiling we can expect from the algorithms, e.g., PANPAL returns a coarse view, modeling only a few tiles which match actually persistent ones, the quality of PANDA+’ results substantially varies and MDL4BMF and particularly PRIMP are most often able to identify the persistent interrelations. Yet how do the algorithms relate in their actual cognition of structure and noise, what makes a tile a tile?

Image data allows us to visually inspect the resulting factorizations without the need to specify a numeric measure. We can intuitively assess the attempts to capture relevant sub-structures. However, some preprocessing is required in order to feed $w \times h$ images to the mining algorithms. We employ a standard representation of images: the RGB888 pixel format. Each of the $w \times h$ pixels is represented by 24 bits, using 8 bits per color (red, green and blue). In order to convert an image into a set of transactions, we divide it into blocks (patches) of 4×4 pixels, resulting in a total of $\frac{w}{4} \times \frac{h}{4}$ transactions per image. We adopt this representation from computer vision, where image patches are a standard preprocessing step for raw pixel data (Jarrett et al 2009). Within each block, let $(r, g, b)_{l,k}$ denote the pixel at row l and column k , where $r, g, b \in \{0, 1\}^8$ are the 8-bit binary representation of its red, green and blue color values. We model the concatenation of all 16 pixels within one block as one transaction

$$[(r, g, b)_{1,1}, (r, g, b)_{1,2}, (r, g, b)_{1,3}, (r, g, b)_{1,4}, (r, g, b)_{2,1}, \dots, (r, g, b)_{4,4}] \quad (11)$$

which has a length of $24 \cdot 16 = 384$ bits.

This way, we process two images: an illustration of *Alice* in Wonderland (Fig. 11) and a selection of “aliens” from the classic game *Space Invaders* (Fig. 12). We select Alice because the image contains multiple connected areas, each representing a reasonable substructure, i.e., hair, face, dress, arm and background. In return, the Space Invaders image contains multiple patterns in terms of color and shape, but the components are clearly spatially separable.

The original Alice image, as well as reconstructions $\theta(XY)$ and the top-4 tiles generated by NASSAU, MDL4BMF, PANDA+, PANPAL and PRIMP, are depicted in Fig. 11. Clearly, only PANDA+ and PRIMP select patterns, i.e., blocks of pixels which provide a reasonable reconstruction of the original image. PANPAL’s tendency to underestimate the rank (choosing only three factors) becomes apparent

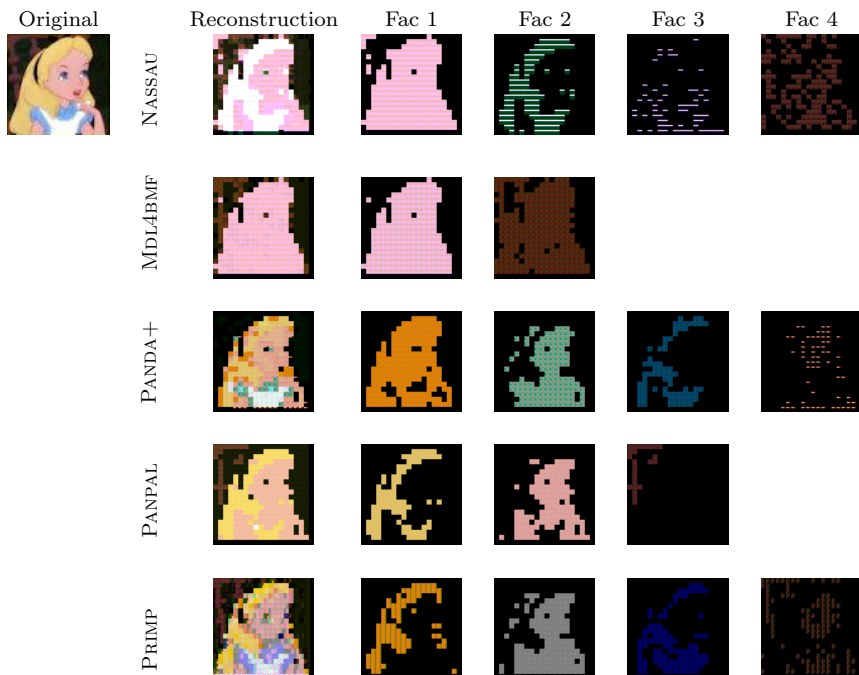


Fig. 11: Reconstructions of the Alice image and postprocessed top-4 tiles. Best viewed in color.

here again. Regarding the figured structures, PANDA+, PANPAL and PRIMP discover a hair-related substructure, where the one found by PRIMP has the most distinctive contours, and PANDA+, PANPAL and PRIMP identify a face-related structure. The reconstructions and factors found by NASSAU and MDL4BMF are not easy to interpret without knowledge of the original image.

Reconstruction results and top-4 patterns of the Space Invaders image are shown in Fig. 12. All methods reconstruct at least the shape of the aliens. In terms of color, however, the results diverge. PANDA+ and NASSAU interpret all colors as negative noise effects on the color white; white has a binary representation of 24 ones. PANPAL recovers the yellow color correctly and it extracts the full blue channel from the image—an identical pattern is also detected by PRIMP. PRIMP and MDL4BMF reconstruct all three colors of the original image, yet the reconstruction of MDL4BMF exhibits injections of white blocks. Hence, only PRIMP is capable to reconstruct the color information correctly.

Having a look at derived tiles, the greedy processes of PANDA+ and NASSAU become particularly visible; PANDA+ and NASSAU overload the first factor with all the shape information. The remaining factors reduce the quantitative reconstruction error, but have no deeper interpretation. MDL4BMF tries to model one type of aliens by each tile. Although this would result in a reasonable description of the image, the actual extraction of tiles suffers from the greedy implementation. We can see that, e.g., the first tile captures information about the yellow aliens as well as strayed parts of other aliens. This unfortunate allocation of tiles results in

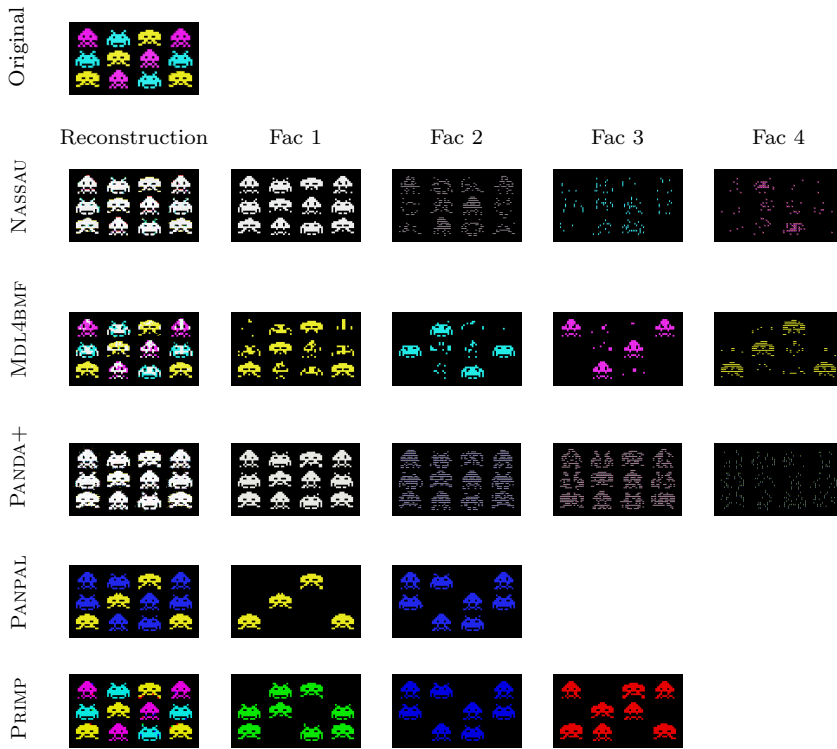


Fig. 12: Reconstructions of the Space Invaders image and the top-4 postprocessed tiles. Best viewed in color.

the injection of white blocks in the reconstruction image. PANPAL clearly separates yellow and blue aliens but interprets differences from the color blue to purple and to turquoise as noise. Finally, PRIMP separates by its tiles the three basic color channels which are actually used to mix the colors that appear in the original image. Hence, PRIMP achieves the factorization rank that corresponds to the natural amount of color concepts in the image, unlike all other competitors.

The results of this qualitative experiment particularly illustrates the benefits of a non-greedy minimization procedure. Even though PANPAL is often not able to minimize the costs due to an underestimation of the rank, its categorization into tiles always yields interpretable parts.

5 Conclusion

We introduce PAL-TILING, a general framework to compute tilings according to a cost measure based on a theoretically founded numerical optimization technique. Requiring that the cost measure has a smooth relaxed function, which combines the matrix factorization error with a regularizing function, PAL-TILING minimizes the relaxed objective under convergence guarantees. To simulate the minimization subject to the constraint that the matrices are binary, we derive a closed form of the

proximal mapping with respect to a function which penalizes non-binary values. A thresholding to binary values according to the actual cost measure enables an automatic determination of the factorization rank.

Aiming at the robust identification of tilings in presence of various noise distributions, we consider two cost measures in this framework which defines two tiling algorithms. The first algorithm uses a simple $L1$ -norm regularization on the factor matrices and is called PANPAL. The second minimizes the MDL-description length of the encoding by code tables as known from KRIMP (Siebes et al 2006). Foregoing the heuristics in computing the usage of codes, we extend the application of this encoding from pattern mining to Boolean matrix factorization and derive an upper bound which induces the relaxed objective. We refer to this instance of PAL-TILING as PRIMP.

Our experiments on synthetically generated datasets show that the quality of competing algorithms PANDA+, MDL4BMF and NASSAU is sensitive towards multiple data generation parameters. The first of the two newly introduced algorithms, PANPAL, regularly underestimates the true factorization rank. We have seen that this property can be beneficial in settings with large, overlapping tiles which induce dense datasets (cf. Fig. 9). In all other settings, the second algorithm PRIMP is able to detect the underlying structure, regardless of the considered distribution of noise or variations the factorization rank (cf. Figs. 5-8).

A comparison of cost measures on real-world datasets show that PRIMP also most often achieves lowest costs (cf. Table 5). With experiments based on images, we visualize the derived tiles under presence of ambiguous tiling structures and special noise distributions (cf. Figs. 11 and 12). The quality of the reconstruction by established algorithms varies considerably between both images. On the contrary, PANPAL and PRIMP provide solid representations of the original images. The extracted factors reveal a parts-based decomposition of the data (as known from non-negative matrix factorizations), which allows for interpretation of the results. In the Space Invaders image (cf. Fig. 12), PANPAL partitions the space invaders into those with a non-zero blue component in their color (rank-1 factorization 2) and those with a zero blue component in their color (rank-1 factorization 1). On the other hand, PRIMP divides the space invaders by the primary colors they contain (repeating each space invader exactly twice, hence finding structure in the data too, albeit a different structure from the one found by PANPAL). From the Alice image (cf. Fig. 11) particularly PRIMP manages to extract coherent factors representing the hair (rank-1 factorization 1) and the face (rank-1 factorization 4).

The implementation of other popular cost measures, e.g., the Typed XOR DtM, is possible in PAL-TILING and a topic of future research. Furthermore, the application of other penalizing functions ϕ is possible if the corresponding prox-operator can be derived. An analysis of the synergy between the penalizing function, the cost-measure and the thereby derived Boolean Matrix Factorization has the potential to show how the structure from arbitrary binary datasets can be robustly identified.

Acknowledgements Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, projects A1 and C1 <http://sfb876.tu-dortmund.de>.

Furthermore, we thank Jilles Vreeken and Sanjar Karaev for their support in the execution of experiments and useful remarks.

References

- Bauchhage C (2015) k-means clustering is matrix factorization. arXiv preprint arXiv:151207548
- Bolte J, Sabach S, Teboulle M (2014) Proximal alternating linearized minimization for non-convex and nonsmooth problems. *Mathematical Programming* 146(1-2):459–494
- Cover T, Thomas J (2006) *Elements of information theory*. Wiley-Interscience
- De Bie T (2011) Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Mining and Knowledge Discovery* 23(3):407–446
- Ding C, Li T, Peng W, Park H (2006) Orthogonal nonnegative matrix t-factorizations for clustering. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp 126–135
- Ding CH, He X, Simon HD (2005) On the equivalence of nonnegative matrix factorization and spectral clustering. In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*, SIAM, vol 5, pp 606–610
- Geerts F, Goethals B, Mielikäinen T (2004) Tiling databases. In: *International Conference on Discovery Science*, Springer, pp 278–289
- Grünwald P (2007) *Minimum Description Length Principle*. MIT press, Cambridge, MA
- Hess S, Piatkowski N, Morik K (2014) Shrimp: Descriptive patterns in a tree. In: *Proceedings of the 16th LWA Workshops: KDML, IR and FGWM.*, pp 181–192
- Jarrett K, Kavukcuoglu K, Ranzato M, LeCun Y (2009) What is the best multi-stage architecture for object recognition? In: *IEEE 12th International Conference on Computer Vision*, IEEE Computer Society, pp 2146–2153
- Karaev S, Miettinen P, Vreeken J (2015) Getting to know the unknown unknowns: Destructive-noise resistant boolean matrix factorization. In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*, SIAM, pp 325–333
- Kontonasis KN, De Bie T (2010) An information-theoretic approach to finding informative noisy tiles in binary databases. In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*, SIAM, pp 153–164
- Kuhn HW (1955) The hungarian method for the assignment problem. *Naval research logistics quarterly* 2(1-2):83–97
- Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791
- Lee DD, Seung HS (2001) Algorithms for non-negative matrix factorization. In: *Advances in neural information processing systems*, pp 556–562
- Li PVM (1997) *An Introduction to Kolmogorov Complexity and Its Applications*. Springer
- Li T (2005) A general model for clustering binary data. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, ACM, pp 188–197
- Li T, Ding C (2006) The relationships among various nonnegative matrix factorization methods for clustering. In: *International Conference on Data Mining (ICDM)*, IEEE, pp 362–371
- Lucchese C, Orlando S, Perego R (2010) Mining top-k patterns from binary datasets in presence of noise. In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*, SIAM, vol 10, pp 165–176
- Lucchese C, Orlando S, Perego R (2014) A unifying framework for mining approximate top-k binary patterns. *Transactions on Knowledge and Data Engineering* 26(12):2900–2913
- Miettinen P (2015) Generalized matrix factorizations as a unifying framework for pattern set mining: Complexity beyond blocks. In: *Machine Learning and Knowledge Discovery in Databases*, Springer International Publishing, pp 36–52
- Miettinen P, Vreeken J (2014) Mdl4bmf: Minimum description length for boolean matrix factorization. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8(4):18:1–18:31
- Miettinen P, Mielikainen T, Gionis A, Das G, Mannila H (2008) The discrete basis problem. *IEEE Transactions on Knowledge and Data Engineering* 20(10):1348–1362
- Paatero P, Tapper U (1994) Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics* 5(2):111–126
- Parikh N, Boyd S (2014) Proximal algorithms. *Foundations and Trends in Optimization* 1(3)
- Rissanen J (1978) Modeling by shortest data description. *Automatica* 14:465–471
- Siebes A, Kersten R (2011) A structure function for transaction data. In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*, SIAM, pp 558–569
- Siebes A, Vreeken J, van Leeuwen M (2006) Item sets that compress. In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*, SIAM, vol 6, pp 393–404

- Smets K, Vreeken J (2012) Slim: Directly mining descriptive patterns. In: Proceedings of the SIAM International Conference on Data Mining (SDM), SIAM, pp 236–247
- Tatti N, Vreeken J (2012) Comparing apples and oranges: measuring differences between exploratory data mining results. *Data Mining and Knowledge Discovery* 25(2):173–207
- Van Leeuwen M, Siebes A (2008) Streamkrimp: Detecting change in data streams. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp 672–687
- Vreeken J, Van Leeuwen M, Siebes A (2011) Krimp: mining itemsets that compress. *Data Mining and Knowledge Discovery* 23(1):169–214
- Wang YX, Zhang YJ (2013) Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering* 25(6):1336–1353
- Xiang Y, Jin R, Fuhry D, Dragan FF (2011) Summarizing transactional databases with overlapped hyperrectangles. *Data Mining and Knowledge Discovery* 23(2):215–251
- Zhang Z, Ding C, Li T, Zhang X (2007) Binary matrix factorization with applications. In: Seventh IEEE International Conference on Data Mining (ICDM), pp 391–400
- Zimek A, Vreeken J (2013) The blind men and the elephant: On meeting the problem of multiple truths in data from clustering and pattern mining perspectives. *Machine Learning* 98(1-2):121–155

A Derivation of the Proximal Operator

Theorem 1 *Let $\alpha > 0$ and $\phi(X) = \sum_{i,j} \Lambda(X_{ij})$ for $X \in \mathbb{R}^{m \times n}$. The proximal operator of $\alpha\phi$ maps the matrix X to the matrix $\text{prox}_{\alpha\phi}(X) = A \in [0, 1]^{m \times n}$ defined by $A_{ji} = \text{prox}_{\alpha\Lambda}(X_{ji})$, where for $x \in \mathbb{R}$ it holds that*

$$\text{prox}_{\alpha\Lambda}(x) = \begin{cases} \max\{0, x - 2\alpha\} & x \leq 0.5 \\ \min\{1, x + 2\alpha\} & x > 0.5. \end{cases} \quad (9)$$

Proof Let $\alpha > 0$, $X \in \mathbb{R}^{m \times n}$ for some $m, n \in \mathbb{N}$ and $A = \text{prox}_{\alpha\phi}(X)$. The function ϕ is fully separable across all matrix entries. In this case, the proximal operator can be applied entry-wise to the composing scalar functions (Parikh and Boyd 2014), i.e., $A_{ji} = \text{prox}_{\alpha\Lambda}(X_{ji})$. It remains to derive the proximal mapping of Λ (Eq. (9)).

The proximal operator reduces to Euclidean projection if the argument lies outside of the function’s domain (Parikh and Boyd 2014) and it follows that

$$\text{prox}_{\alpha\Lambda}(x) = \theta(x) \text{ if } x \notin [0, 1].$$

For $x \in [0, 1]$ holds $\Lambda(x) = -|1 - 2x| + 1$ and

$$\begin{aligned} \text{prox}_{\alpha\Lambda}(x) &= \arg \min_{x^* \in \mathbb{R}} \left\{ \frac{1}{2}(x - x^*)^2 - \alpha|1 - 2x^*| + 1\alpha \right\} \\ &= \arg \min_{x^* \in \mathbb{R}} \left\{ \underbrace{(x - x^*)^2 - 2\alpha|1 - 2x^*| + (2\alpha)^2}_{=g(x^*; x, \alpha)} \right\}, \end{aligned}$$

where g is derived by a multiplication and addition of constants, such that the minimum can easily be derived by completing the square.

$$\begin{aligned} g(x^*; x, \alpha) &= \begin{cases} (x - x^*)^2 - 2\alpha(1 - 2x^*) + (2\alpha)^2 & x^* \leq 0.5 \\ (x - x^*)^2 + 2\alpha(1 - 2x^*) + (2\alpha)^2 & x^* > 0.5 \end{cases} \\ &= \begin{cases} (x^* - (x - 2\alpha))^2 - 2\alpha(1 - 2x) & x^* \leq 0.5 \\ (x^* - (x + 2\alpha))^2 + 2\alpha(1 - 2x) & x^* > 0.5 \end{cases}. \end{aligned}$$

The function g is a continuous piecewise quadratic function which attains its global minimum at the minimum of one of the two quadratic functions, i.e.,

$$\arg \min_{x^* \in \mathbb{R}} g(x^*; x, \alpha) \in \{x - 2\alpha \mid x \leq 0.5 + 2\alpha\} \cup \{x + 2\alpha \mid x > 0.5 - 2\alpha\}.$$

A function value comparison in the intersecting domain $x \in (0.5 - 2\alpha, 0.5 + 2\alpha]$ yields that

$$g(x - 2\alpha; x, \alpha) = -2\alpha(1 - 2x) \leq g(x + 2\alpha; x, \alpha) = 2\alpha(1 - 2x) \Leftrightarrow x \leq 0.5$$

□

B Krimp's Encoding as Matrix Factorization

Lemma 1 *Let D be a data matrix. For any code table CT and its cover function there exists a Boolean matrix factorization $D = \theta(YX^T) + N$ such that non-singleton patterns in CT are mirrored in X and the cover function is reflected by Y . The description lengths correspond to each other, such that*

$$L_{CT}(D, CT) = f_{CT}(X, Y, D) = f_{CT}^D(X, Y, D) + f_{CT}^M(X, Y, D),$$

where the functions returning the model and the data description size are given as

$$\begin{aligned} f_{CT}^D(X, Y, D) &= - \sum_{s=1}^r |Y_{\cdot s}| \cdot \log(p_s) - \sum_{i=1}^n |N_{\cdot i}| \cdot \log(p_{r+i}) \\ &= L_{CT}^D(D, CT) \\ f_{CT}^M(X, Y, D) &= \sum_{s: |Y_{\cdot s}| > 0} (X_{\cdot s}^T c - \log(p_s)) + \sum_{i: |N_{\cdot i}| > 0} (c_i - \log(p_{r+i})) \\ &= L_{CT}^M(CT). \end{aligned}$$

The probabilities p_s and p_{r+i} indicate the relational usage of non-singleton patterns $X_{\cdot s}$ and singletons $\{i\}$,

$$p_s = \frac{|Y_{\cdot s}|}{|Y| + |N|}, \quad p_{r+i} = \frac{|N_{\cdot i}|}{|Y| + |N|}.$$

We denote with $c \in \mathbb{R}_+^n$ the vector of standard code lengths for each item, i.e.,

$$c_i = - \log \left(\frac{|D_{\cdot i}|}{|D|} \right).$$

Proof Let D be a data matrix, $CT = \{(X_\sigma, C_\sigma) \mid 1 \leq \sigma \leq \tau\}$ a τ -element code table and cover the cover function. Let r be the number of non-singleton patterns in CT and assume w.l.o.g. that CT is indexed such that these non-singleton patterns have an index $1 \leq \sigma \leq r$. We construct the pattern matrix $X \in \{0, 1\}^{n \times r}$ and usage matrix $Y \in \{0, 1\}^{m \times r}$ such that for $1 \leq \sigma \leq r$ it holds that

$$\begin{aligned} X_{i\sigma} &= 1 \Leftrightarrow i \in X_\sigma \\ Y_{j\sigma} &= 1 \Leftrightarrow X_\sigma \in \text{cover}(CT, D_j). \end{aligned}$$

The Boolean product $\theta(YX^T)$ indicates the entries of D which are covered by non-singleton patterns of CT . That implies that ones in the noise matrix $N = D - \theta(YX^T)$ are covered by singletons, it holds that

$$N_{ji} \neq 0 \Leftrightarrow i \in \text{cover}(CT, D_j).$$

The usage of a non-singleton pattern X_σ is then computed as

$$\begin{aligned} \text{usage}(X_\sigma) &= |\{X_\sigma \in \text{cover}(CT, D_j.) \mid j \in \mathcal{T}\}| \\ &= |\{Y_{j\sigma} = 1 \mid j \in \mathcal{T}\}| \\ &= |Y_{\cdot \sigma}|, \end{aligned}$$

and correspondingly it follows that $\text{usage}(\{i\}) = |N_{\cdot i}|$. The calculation of the probabilities p_σ for $1 \leq \sigma \leq r + n$ is directly obtained by inserting this usage calculation in the definition of code-usage-probabilities of Eq. (2). Likewise follow the functions f_{CT}^M and f_{CT}^D from the definition of the description sizes L_{CT}^M and L_{CT}^D . □

C Bounding the Description Length of Code Tables

Lemma 1 Let (a_s) be a finite sequence of r non-negative scalars such that $S_r = \sum_{s=1}^r a_s > 0$. The function $g: [0, \infty) \rightarrow [0, \infty)$ defined by

$$g(x; a_1, \dots, a_r, S_r) = - \sum_{s=1}^r (a_s + x) \log \left(\frac{a_s + x}{S_r + rx} \right)$$

is monotonically increasing in x .

Proof W.l.o.g., let $a_1, \dots, a_{r_0} > 0$ and $a_{r_0+1}, \dots, a_r = 0$ for some $r_0 \in \mathbb{N}$. We rewrite the function g as

$$g(x; a_1, \dots, a_r, S_r) = g(x; a_1, \dots, a_{r_0}, S_r) + g(x; a_{r_0+1}, \dots, a_r, S_r)$$

and show that each of the subfunctions is monotonically increasing. The first subfunction is differentiable and its derivative is non-negative

$$\begin{aligned} \frac{d}{dx} g(x; a_1, \dots, a_{r_0}, S_r) &= - \sum_{s=1}^{r_0} \left(\log \left(\frac{a_s + x}{S_r + rx} \right) + (a_s + x) \frac{S_r + rx}{a_s + x} \frac{S_r + rx - r(a_s + x)}{(S_r + rx)^2} \right) \\ &= - \sum_{s=1}^{r_0} \log \left(\frac{a_s + x}{S_r + rx} \right) + \sum_{s=1}^{r_0} \frac{S_r - ra_s}{S_r + rx} \\ &= - \sum_{s=1}^{r_0} \log \left(\frac{a_s + x}{S_r + rx} \right) \geq 0. \end{aligned}$$

The second subfunction is monotonically increasing, since for $a_s = 0$ and all $x \geq 0$ it holds that

$$a_s \log \left(\frac{a_s}{S_r} \right) = 0 \leq -(a_s + x) \log \left(\frac{a_s + x}{S_r + rx} \right).$$

□

Theorem 2 Given binary matrices X and Y and $\mu = 1 + \log(n)$, it holds that

$$f_{\text{CT}}^D(X, Y, D) \leq \mu \|D - YX^T\|^2 - \sum_{s=1}^r (|Y_{\cdot s}| + 1) \log \left(\frac{|Y_{\cdot s}| + 1}{|Y| + r} \right) + |Y| \quad (10)$$

Proof We recall that the description size of the data is computed by

$$f_{\text{CT}}^D(X, Y, D) = \underbrace{- \sum_{s=1}^r |Y_{\cdot s}| \cdot \log \left(\frac{|Y_{\cdot s}|}{|Y| + |N|} \right)}_{=f_1(X, Y, D)} - \underbrace{\sum_{i=1}^n |N_{\cdot i}| \cdot \log \left(\frac{|N_{\cdot i}|}{|N| + |Y|} \right)}_{=f_2(X, Y, D)}.$$

Applying the logarithmic properties, we rewrite the first sum

$$\begin{aligned} f_1(X, Y, D) &= - \sum_{s=1}^r |Y_{\cdot s}| \log \left(\frac{|Y_{\cdot s}|}{|Y|} \frac{|Y|}{|Y| + |N|} \right) \\ &= - \sum_{s=1}^r |Y_{\cdot s}| \log \left(\frac{|Y_{\cdot s}|}{|Y|} \right) + \sum_{s=1}^r |Y_{\cdot s}| \log \left(\frac{|Y| + |N|}{|Y|} \right) \\ &= g(0; |Y_{\cdot 1}|, \dots, |Y_{\cdot r}|, |Y|) + |Y| \log \left(1 + \frac{|N|}{|Y|} \right). \end{aligned}$$

It follows from the monotonicity of g (Lemma 1) and the logarithm inequality ($\log(1+x) \leq x, \forall x \geq 0$) that f_1 is upper bounded by

$$f_1(X, Y, D) \leq -\sum_{s=1}^r (|Y_{\cdot s}| + 1) \log \left(\frac{|Y_{\cdot s}| + 1}{|Y| + r} \right) + |N|.$$

The second term f_2 can be transformed into

$$\begin{aligned} f_2(X, Y, D) &= -\sum_{i=1}^n |N_{\cdot i}| \cdot \log(|N_{\cdot i}|) + \sum_{i=1}^n |N_{\cdot i}| \cdot \log(|N| + |Y|) \\ &= \sum_{i=1}^n |N_{\cdot i}| \log \frac{1}{|N_{\cdot i}|} + |N| \log(|N| + |Y|). \end{aligned}$$

Subsequently, we show $f_2(X, Y, D) \leq |N| \log(n) + |Y|$. This inequality trivially holds if $|N| = 0$. Otherwise, we apply Jensen's inequality to the concave logarithm function

$$|N| \sum_{i=1}^n \frac{|N_{\cdot i}|}{|N|} \log \frac{1}{|N_{\cdot i}|} \leq |N| \log \left(\frac{n}{|N|} \right).$$

and obtain

$$\begin{aligned} f_2(X, Y, D) &\leq |N| \log \left(\frac{n}{|N|} \right) + |N| \log(|N| + |Y|) \\ &= |N| \log(n) + |N| \log \left(1 + \frac{|Y|}{|N|} \right) \\ &\leq |N| \log(n) + |Y|, \end{aligned}$$

where the last equality again follows from the logarithm inequality. We derive the final inequality by

$$\begin{aligned} f_{\text{CT}}^D(X, Y, D) &= f_1(X, Y, D) + f_2(X, Y, D) \\ &\leq (1 + \log(n))|N| - \sum_{s=1}^r (|Y_{\cdot s}| + 1) \log \left(\frac{|Y_{\cdot s}| + 1}{|Y| + r} \right) + |Y| \end{aligned}$$

□

D Calculating the Lipschitz Moduli of PRIMP

We study the partial gradients of the regularization term used in PRIMP (Sec. 3.4)

$$\begin{aligned} \nabla_X G(X, Y) &= c(0.5)_s^T \\ \nabla_Y G(X, Y) &= -\frac{1}{2} \left(\log \left(\frac{|Y_{\cdot s}| + 1}{|Y| + r} \right) \right)_{j_s} + (0.5)_{j_s}. \end{aligned}$$

The partial gradient with respect to X is constant and has a Lipschitz constant of zero. The partial gradient with respect to Y can be written as the sum

$$\nabla_Y G(X, Y) = -\frac{1}{2} \left(\underbrace{(\log(|Y_{\cdot s}| + 1))_{j_s}}_{=A(Y)} - \underbrace{(\log(|Y| + r))_{j_s}}_{=B(Y)} \right) + (0.5)_{j_s}.$$

From the triangle inequality follows that the gradient with respect to Y is Lipschitz continuous with modulus $M_{\nabla_Y G}(X) = \frac{1}{2}(M_A + M_B)$, if the functions A and B are Lipschitz continuous with moduli M_A and M_B :

$$\begin{aligned} \|\nabla_Y G(X, Y) - \nabla_Y G(X, V)\| &= \frac{1}{2} \|A(Y) - A(V) + B(Y) - B(V)\| \\ &\leq \frac{1}{2} \|A(Y) - A(V)\| + \|B(Y) - B(V)\| \\ &\leq \frac{M_A + M_B}{2} \|Y - V\|. \end{aligned}$$

The one-dimensional function $x \mapsto \log(x + \delta)$, $x \in \mathbb{R}_+$ is for any $\delta > 0$ Lipschitz continuous with modulus δ^{-1} . This can be easily derived by the mean value theorem and the bound

$$\frac{d}{dx} \log(x + \delta) = \frac{1}{x + \delta} \leq \frac{1}{\delta}$$

for all $x \geq 0$. We show with the following equations, that $M_A = M_B = m$. For improved readability, we use the squared Lipschitz inequality, i.e.,

$$\begin{aligned} \|A(Y) - A(V)\|^2 &= \sum_{s,j} (\log(|Y_{.s}| + 1) - \log(|V_{.s}| + 1))^2 \\ &= m \sum_{s=1}^r (\log(|Y_{.s}| + 1) - \log(|V_{.s}| + 1))^2 \\ &\leq m \sum_{s=1}^r (|Y_{.s}| - |V_{.s}|)^2 \end{aligned} \tag{12}$$

$$\begin{aligned} &= m \sum_{s=1}^r \left(\sum_{j=1}^m (Y_{js} - V_{js}) \right)^2 \\ &\leq m^2 \sum_{s,j} (Y_{js} - V_{js})^2 = m^2 \|Y - V\|^2, \end{aligned} \tag{13}$$

where Eq. (12) follows from the Lipschitz continuity of the logarithmic function as discussed above for $\delta = 1$ and Eq. (13) follows from the Cauchy-Schwarz inequality. Similar steps yield the Lipschitz modulus of B ,

$$\begin{aligned} \|B(Y) - B(V)\|^2 &= \sum_{s,j} (\log(|Y| + r) - \log(|V| + r))^2 \\ &= mr (\log(|Y| + r) - \log(|V| + r))^2 \\ &\leq \frac{mr}{r^2} (|Y| - |V|)^2 \\ &= \frac{m}{r} \left(\sum_{s,j} (Y_{js} - V_{js}) \right)^2 \\ &\leq m^2 \sum_{s,j} (Y_{js} - V_{js})^2. \end{aligned}$$

We conclude that the Lipschitz moduli of the gradients are given as

$$M_{\nabla_X G}(Y) = 0 \quad M_{\nabla_Y G}(X) = m.$$