Georges Miranda Spyrides

Binary Matrix Factorization Post-processing and Applications

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Informática.

Advisor :	Prof. Hélio Côrtes Vieira Lopes
Co-advisor:	Prof. Marcus Vinicius Soledade Poggi de Aragão

Rio de Janeiro
August 2023

Georges Miranda Spyrides

Binary Matrix Factorization Post-processing and Applications

Thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Informática. Approved by the Examination Committee:

Rio de Janeiro, August 24th, 2023

Georges Miranda Spyrides

Graduado em engenharia de produção pela Universidade Federal do Rio de Janeiro e mestre em Informática pela Pontifícia Universidade Católica do Rio de Janeiro.

Bibliographic data

Spyrides, Georges

Binary Matrix Factorization Post-processing and Applications / Georges Miranda Spyrides; advisor: Hélio Côrtes Vieira Lopes; co-advisor: Marcus Vinicius Soledade Poggi de Aragão. – 2023.

112 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. Informática – Teses. 2. Fatoração de Matrizes Binárias. 3. Fatoração de Matrizes Não Negativas. 4. Mineração de Processos. I. Lopes, Hélio. II. Poggi, Marcus. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

A Pedrinho e Helen por um amor sem fim. A meus pais e irmã pela força
para voltar para o meu rumo.

Acknowledgments

To my advisors Professors Hélio Lopes and Marcus Poggi for the encouragement and partnership in conducting this work.

To CNPq and PUC-Rio, for the grants provided, without which this work could not have been accomplished.

To Helen Cristina and Pedro, for all the love in the world and for the patience for the time I invested in this thesis.

To my parents and sister, for the education, attention, and care at all times.

To my friends and colleagues from PUC-Rio. It's a pleasure to study and work with people who are so capable and so generous with their own time. I would especially like to thank Jonatas Grosman for helping me both directly and indirectly in this work.

To the professors who participated in the Examination Committee.

To all friends and family who in one way or another encouraged or helped me.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Spyrides, Georges; Lopes, Hélio (Advisor); Poggi, Marcus (Co-Advisor). **Binary Matrix Factorization Post-processing and Applications**. Rio de Janeiro, 2023. 112p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Novel methods for matrix factorization introduce constraints to the decomposed matrices, allowing for unique kinds of analysis. One significant modification is the binary matrix factorization for binary matrices. This technique can reveal common subsets and mixing of subsets, making it useful in a variety of applications, such as market basket analysis, topic modeling, and recommendation systems. Despite the advantages, current approaches face a trade-off between accuracy, scalability, and explainability. While gradient descent-based methods are scalable, they yield high reconstruction errors when thresholded for binary matrices. Conversely, heuristic methods are not scalable. To overcome this, this thesis propose a post-processing procedure for discretizing matrices obtained by gradient descent. This novel approach recovers the reconstruction error post-thresholding and successfully processes larger matrices within a reasonable timeframe. We apply this technique to many applications including a novel pipeline for discovering and visualizing patterns in petrochemical batch processes.

Keywords

Binary Matrix Factorization; Non-negative matrix factorization; Process Mining.

Resumo

Spyrides, Georges; Lopes, Hélio; Poggi, Marcus. **Pós-processamento de Fatoração Binária de Matrizes e Aplicações**. Rio de Janeiro, 2023. 112p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Novos métodos de fatoração de matrizes introduzem restrições às matrizes decompostas, permitindo tipos únicos de análise. Uma modificação significativa é a fatoração de matrizes binárias para matrizes binárias. Esta técnica pode revelar subconjuntos comuns e mistura de subconjuntos, tornando-a útil em uma variedade de aplicações, como análise de cesta de mercado, modelagem de tópicos e sistemas de recomendação. Apesar das vantagens, as abordagens atuais enfrentam um trade-off entre precisão, escalabilidade e explicabilidade. Enquanto os métodos baseados em gradiente descendente são escaláveis, eles geram altos erros de reconstrução quando limitados para matrizes binárias. Por outro lado, os métodos heurísticos não são escaláveis. Para superar isso, essa tese propõe um procedimento de pós-processamento para discretizar matrizes obtidas por gradiente descendente. Esta nova abordagem recupera o erro de reconstrução após a limitação e processa com sucesso matrizes maiores dentro de um prazo razoável. Testamos esta técnica a muitas aplicações, incluindo um novo pipeline para descobrir e visualizar padrões em processos petroquímicos em batelada.

Palavras-chave

Fatoração de Matrizes Binárias; Fatoração de Matrizes Não Negativas; Mineração de Processos.

Table of contents

List of figures

List of tables

List of algorithms

List of Abreviations

BackColumn – *Backtracking algorithm for the column sub-problem*

BackDisc – *Backtracking algorithm for BMF Discretization*

BMF – *Binary Matrix Factorization*

BoolMM – *Boolean Matrix Multiplication*

OF – *Objective Function*

PCA – *Principal Component Analysis*

SavGol – *Savitzky-Golay polynomial filter*

SVD – *Singular Value Decomposition*

If you can dream—and not make dreams
your master;
If you can think—and not make thoughts
your aim; [...]
Yours is the Earth and everything that's in it,
And—which is more—you'll be a Man, my
son!

Rudyard Kipling, *If.*

1 Introduction

Recent progress in smartphones, internet-of-things devices, and their use of sensors, precision medicine, and social media have brought a vast range of datasets to the public. Frequently the data is about the relationship between consumers and products, entities to another entity, and observations to some features. Even in unusual settings such as text mining, we are used to transforming data to vector representations, such as the count of specific words (bag-of-words) or the presence of words in binary vectors, often called one-hot encoding.

An approach to discovering these datasets' underlying structure is matrix factorization. The most common method is the principal component analysis (PCA). The PCA is an algorithm for capturing greedily the variance of a given matrix, constructing the best low-rank approximation rank by rank. Thus the PCA reconstruction produces the best real-valued approximation for any given matrix (ECKART; YOUNG, 1936) under the Frobenius or 2-norm. This is a result we later use for comparison.

Let A be a real-valued matrix and G be the desired rank for an approximation. The PCA algorithm returns orthonormal matrices U and V, and a diagonal matrix Σ that, when multiplied as in $U \cdot \Sigma \cdot V^T = A_{\text{approx}}$. This decomposition can be used for signal/image compression, embedding observations in a lower dimensional and orthogonalized space, enhancing the predictive power of simple machine learning methods such as decision trees, and unsupervised outlier detection, among others (WOLD; ESBENSEN; GELADI, 1987).

Newer methods change the unconstrained setting of the PCA to obtain decomposed matrices with unique properties. Popular methods combine introducing sparseness-inducing mechanisms or relaxing the orthogonality for just searching for independence between components. A particularly powerful constraint is allowing the factorized matrices to just positive entries. The non-negative constraint forces the reconstruction to rely only on basic addition. Therefore, each component must assume parts of the objects expressed in given matrices. Lee and Seung (LEE; SEUNG, 1999) explain this effect in greater detail and how these matrices produce explainable results in real settings.

In an even more constrained setting, authors such as Zhang et al. (ZHANG et al., 2007) (ZHANG et al., 2010) and Miettinen et al. (MIETTI-

NEN et al., 2008) describe a binary matrix factorization for binary matrices. Given a matrix with binary (or boolean) entries $A_{[m \times n]}$, the binary matrix factorization obtains ideally two also binary matrices $W_{[m \times g]}$ and $H_{[g \times n]}$ that when multiplied together are a rank-g approximation to A.

Binary matrices can be used to encode a myriad of problems. Binary vectors can be used to represent the presence of an item in a set. The resulting matrices of the decomposition reveal common subsets and mixing of subsets to approximate each original set in the decomposed matrix. Applications that benefit from this form of interpreting the decomposition are market basket analysis, topic modeling in text mining, microarray gene expression for sample and genes biclustering, any problem using categorical features for many observations, recommendation systems based on discrete features of user behavior, the relationship between entities such as friendships in social networks.

In a more general setting of a table with observations in the rows and binary features in the columns, such as the presence of a specific word in a document, or purchase of an item in one shopping cart, or the presence of a category in a categorical variable, we can interpret of matrix H as a common subset of features and matrix W as a mixing matrix that shows which subsets combined give an approximation to a specific observation. Common subsets of features obtained in matrix H can be used later to decide which products to place together in an actual store or web interface, common words in a topic inside a text corpus. The mixing matrix W can give statistics of the most common subsets and clustering of observations based on common subsets.

In an entity relationship setting such as in friendships in social networks, or movies watched on some streaming platform, we imagine that the given matrix A represents with binary entries the relationship of entities in the rows (frequently users) to another entity in the columns (such as movies watched, product purchased, celebrity liked, another user followed). This setting can be modeled as a bipartite graph. The binary matrix factorization can be interpreted as detecting communities (bicliques) in these bipartite settings. For instance, almost all inside a group users in the rows in which the first column of W is equal to one have watched almost all movies in the group where the first row of H have entries equal to one. Thus one could find which groups of users have watched which group of movies and strongly recommend missing entries to users based on their close peer's preferences.

Therefore, there is a great benefit to analyzing the structure of binary matrices through the lens of binary matrix factorization. What limits its wide adoption in a real setting is the practical trade-off of current approaches.

Approaches that can handle real-world sized problems, such as those described in Zhang et al. (ZHANG et al., 2007) and implemented by Zitnik and Zupan in their software package Nimfa (ZITNIK; ZUPAN, 2012), are based on gradient descent methods, and their response is almost binary but still fractional.

These matrices are obtained by gradient descent; when thresholded to obtain truly binary matrices, the reconstruction errors increase so that the reconstructed matrices no longer represent the original matrices. For natural settings where interpreting the problem as subsets of elements are lost to this approach. Other approaches, such as those described by Miettinen et al. (MIETTINEN et al., 2008), are truly binary but do not scale outside of matrices in a few hundred rows and columns for ranks G over 20.

Our contribution is a post-processing procedure for discretizing matrices obtained by gradient descent. This algorithm recovers the reconstruction error after the thresholding in our experimental setting. We were able to process in reasonable time matrices with thousands of rows and columns and a rank of up to 50.

1.1 The outer product view

To introduce the terminology used throughout this thesis and to build some intuition, we review the matrix multiplication through the lens of the outer product. The general public is more used to the inner product of matrices to calculate each position of the resulting matrix. The outer product view produces rank-one matrices, or "components" as we call them, and they are summed to produce the final multiplication. The components view makes some inner patterns more apparent for practical application.

The multiplication of binary matrices produces an interesting pattern. In particular when analyzed through the lens of the outer product. In the example below in equation 1-2, we call the first two matrices being multiplied as W and H, and the final product we call A. Notice that usually, we think of matrix multiplication through the lens of the inner product (rows times columns) instead of the outer product (columns times rows). To make it clear, we highlight the columns of W in different colors and the rows of H in different colors.

The outer product of two vectors, w_{blue} (a column vector) and h_{blue} (a row vector), results in a rank-one matrix. If we take the columns of W and rows of H as these vectors, the product $W \cdot H$ results in the sum of several rank-one matrices, where each rank-one matrix is the outer product of a column of W and a row of H. Each rank-one matrix captures a specific feature present in the resulting matrix A.

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1-1}$$

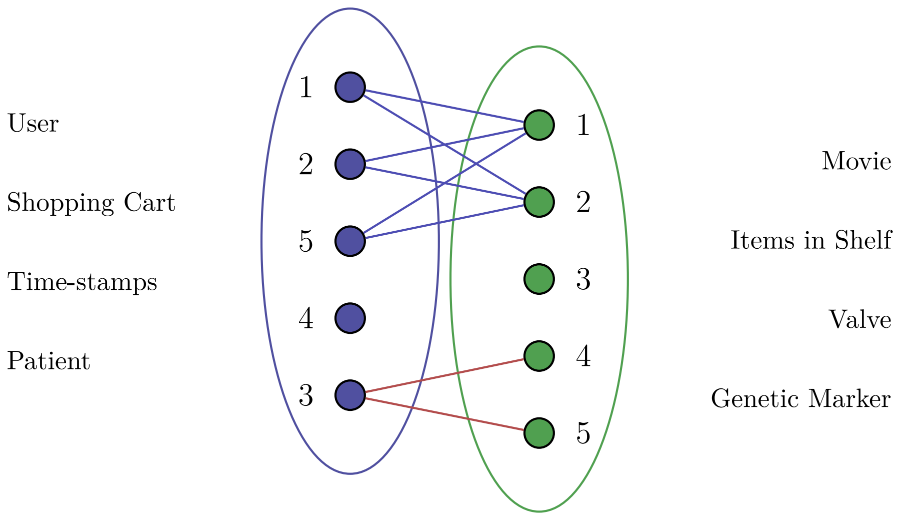
$$= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{1-2}$$

Suppose that matrix A represents in its rows users in a streaming platform, and the columns of matrix A represent the movies in this platform. Each entry carries information about whether the user in row m has seen the movie in column n. Therefore, the entry a_{mn} is one if the user m has seen movie n.

We can notice that users $\{1, 2, 5\}$ have watched movies $\{1, 2\}$, forming the pattern highlighted in blue. Also, user 3 has watched movies $\{4, 5\}$, composing the pattern highlighted in orange. If we go back to the matrices W and H, this description of groups of users who have watched the same movies appear in each pair of column of W associated with the row of H. These patterns are useful for analyzing a situation with many users and movies. An analyst could figure out what users to advertise a new release based on groups and preferences, or could figure a way of categorizing movies in his base.

Thus, it is interesting to make the reverse operation, that means, given a matrix A carrying the relationship of users and movies to decompose them into these patterns, or components or bicliques as we shall call them from now on.

The matrix A could represent other things. It could represent in the columns items available in a supermarket, and in the rows each shopping basket. Then, matrix H would represent groups of items usually bought together in each row and in matrix W the groups of items each user bought. One could also cluster together users that purchased the same group observing the columns of W.



To generalize, we can think, as in the picture above, that in the relationship between two entities we are seeking for groups of observations of entity in the left that have relationship with all obeservations in a group from the entity in the right. Thus, we seek biclusters in a bipartite setting.

This proposed reverse operation is the objective of the binary factorization and the main subject of this thesis. The structure revealed in matrices W and H shows relationship patterns between two entities.

Let us analyze one more example in 1-4. Here the multiplication of W and H produces components that over-cover the central position of the matrix (highlighted with pink).

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1-3}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1-4}$$

This effect of over-covering a position has some interesting consequences. The multiplication of binary matrices can produce non-binary matrices. Thus if the resulting matrix A had a one in the central position instead of two, and we were to decompose the matrix, the decomposition into W and H wouldn't be perfect because the reconstruction would have these two in the central position. Thus the factorization doesn't produce an ideal reconstruction. And the quality metric of the reconstruction has to consider the cost of over-covering a position with a 1 or a 0, cost of under-covering a position with a 1.

Our proposed algorithm pipeline can mine large matrices, with a comparatively large number of components with a lower error than other matrices

1.2 Objectives and Scope

This thesis is has aspects unsupervised machine learning, consequently its value is measured on the possibility of applications in many situations. Naturally, some applications will call for an adaptation. We observe a large variability in approaches to this problem and a large variability in evaluation metrics.

Our definition of scope considered that, in practice, binary decompositions have a simultaneous need for accuracy, scalability, and explainability. It is important for factorized matrices to accurately reconstruct the original matrices. Also, it is important to scale the approaches to large matrices and factorizations of larger ranks. More ranks mean more rank-one matrix explaining the solutions, and more patterns mined.

Finally, explainability calls for two qualitative aspects. One of them is the ability to translate binary vectors to sets of items. Therefore the need for strictly binary factors in both factorized matrices W and H. The other aspect is minimizing the ambiguity of causes, which usually manifests as overcovering positions in the reconstruction of the given matrix A.

Solution methods based on gradient descent and alternating least-squares produce near-binary real-valued matrices with high accuracy and highly scalable solutions. However, they suffer from a decision problem. If the application needs an enumeration of items in each component, the usual route is to consider items above a certain threshold t. In the literature and our tests, we observe that any approach created to this thresholding profoundly deteriorates the accuracy of the problem.

Heuristic methods usually choose between greedy behavior or an intractable combinatorial search space. Greedy approaches typically find bicliques or dense cores, but they fail to take advantage of factorization to larger ranks. They usually find dense cores and cannot get better past a certain rank. So they find dense patterns and stop early.

Our objective was to answer the following main research question: Is it possible to solve the problem of discretizing near-binary factorizations by recovering the typical explosion in error when thresholding? This way, we wanted to explore the efficiency of gradient descent and alternating least squares methods and discretize them without the usual loss in accuracy of the simple thresholding. The discretization may pave the way for unifying the chasm in the literature between discrete methods and continuous methods.

We had a secondary objective of unifying the continuous-first approaches in the literature with discrete methods. Some of the challenges was proving that a common measure used by the literature has essential flaws. For instance, it does not allow for comparing with continuous factorization such as the PCA, NMF or even continuous approximate versions of BMF. Additionally it hides the effect of overcovering, allowing for algorithms that tend to produce components with large intersections, thus ambiguous answers.

Other secondary objectives were application-related. Our main application to process mining had its own challenges. For instance, we had the goal of discovering how to cluster valves and represent their pattern in time, with understandable representation. Our approach to the binary matrix factorization was essential to achieve this goal, and the crucial step in a larger solution, as shown in chapter 5.1.1.

1.2.1 Our approach and main contribution

We present a method for recovering error for near binary factorizations in a reasonable time, for matrices of tens of thousands of rows and thousands of columns, and for a decomposition rank of up to 50. The recovery is particularly interesting for using gradient-descent methods that scale well in practice and have high accuracy before discretizing. These methods also scale to larger ranks better than the heuristic approaches because they have a top-down construction method for learning the components.

Compared to other approaches, our proposed pipeline has a compelling mix of accuracy, scalability, and explainability.

1.2.2 Measuring the quality of the results

We use the Frobenius relative error using standard matrix multiplication, as opposed to boolean matrix multiplication. The Frobenius norm of a matrix, also known as the Euclidean norm or the 2-norm, is a measure of the "magnitude" or "size" of the matrix.

If A is a matrix with m rows and n columns, and a_{ij} denotes the entry in the ith row and jth column, the Frobenius norm of A is defined as:

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$$

If we want to measure the distance of a matrix A and an approximation by factorized matrices $W \cdot H$, we have to take the norm of the residual difference between them. Finally we normalize the error by norm of the original matrix, to obtain a relative measure. Thus we can calculate the Frobenius norm difference using the following formula:

$$||A - W \cdot H||_F / ||A||_F \tag{1-5}$$

This represents how much of the variance of matrix A was captured by the factorization or is encoded in matrices W and H. The better approximation $W \cdot H$ produces, the closer this error comes to zero.

Some related works use a boolean matrix multiplication that aggregates components with an OR function instead of the sum. Later in the method chapter 3, we present a formal comparison of both metrics. In practice, the boolean matrix factorization is only possible with boolean matrices; our approach post-processes a quasi-binary matrix, so it would be impossible to calculate the error recovery using this metric. Also, we prove in chapter 3 that using the regular matrix multiplication penalizes the effect of overcovering a position with more than one component.

Finally, there is bilinearity of the error in equation 1-5. Additionally, we consider desired matrices W and H to have binary entries, so the optimization landscape of this problem turns into an unconstrained binary quadratic problem (UBQP) which is very hard and doesn't scale well. This optimization landscape is hard to navigate, full of local optima, and many approaches (including ours) are not deterministic to approximate.

Therefore, we compare our results to the Frobenius reconstruction error calculated using the PCA reconstruction. The PCA navigates in a less constrained space that contains the space of binary matrices, so it is guaranteed to produce a better error norm than the BMF. More than that, it is the best reconstruction of a given rank G, a unique and deterministic best. If the PCA processes matrix A into a rank-G factorization, yielding matrices U_g, Σ_g, and V_g^T, their Frobenius norm reconstruction error can be calculated as the following equation.

$$||A - U_g \cdot \Sigma_g \cdot V_g^T||_F / ||A||_F \tag{1-6}$$

So it works as a deterministic and unique lower-bound on every instance. This measure allows us to understand if the instance was even decomposable into G components in the first place.

If we used the boolean matrix multiplication in 1-5, it would not be comparable to this lower bound given by the PCA.

1.3 Work organization

In chapter 1, we present an overview of the area of binary matrix factorization, our objectives, contributions, and how we organized this work.

Then, in Chapter 2, we present the main reference for the work. We divided our reference search into a discussion about the main algorithmic approaches for Binary Factorization and their common applications.

In Chapter 3, we present the full pipeline for our approach. We propose a discretization step at the end of a factorization pipeline. Thus we present the problem formulation and the column subproblem, an example as a motivation to the theoretical work, the main algorithm for solving the column subproblem, and then we discuss how to use the algorithm for the column subproblem back into the factorization pipeline. Also, we briefly compare the Frobenius norm reconstruction error using the regular matrix multiplication and the boolean matrix multiplication present in related works.

Our work presents many different applications and is measured with different error scores. It was worth it to distribute it into three chapters. In Chapter 4, we present three tiny cases that allow us to observe nuances in algorithmic behavior regardless of the quality metric choice. Thus, we compare two algorithms and our pipeline's solution to each of these cases.

In Chapter 5.1.1, we introduce the reader to our main application case: the unsupervised visualization of petrochemical batch processes. Actually, this application was the first motivation for the present work. We show how we have applied the binary matrix factorization along with the principal component analysis to obtain a representation of the operation of valves of a batch process in a petrochemical industry. Also, we present all the visualization of the 11 selected time periods of two refineries in Appendix A.

In Chapter 6, we present four other applications of the algorithm. First, we compare our pipeline to different algorithms using the datasets found in (DESOUKI; RöDER; NGOMO, 2019). Then we apply our pipeline to Gene Expression datasets. This application was first tried by the same author whose algorithm inspired this post-processing pipeline in (ZHANG et al., 2010). The difference in our work is that we apply our pipeline to larger instances and a larger number of them. These instances are available in the portal Gene Expression Omnibus, a database from the U.S.'s National Library of Medicine (EDGAR; DOMRACHEV; LASH, 2002).

Additionally, we apply our pipeline to more general applications of topic modeling and recommendation systems. These applications usually have difficult instances and enrich our discussion of results. We present the detailed results table from the gene expression datasets and the topic modeling in appendix B

Finally, in Chapter 7, we make a final discussion over the main strengths and shortcomings of our approach compared to others and our view of the kinds of applications we believe our approach should work best. In addition, we also present ideas that we want to delve into in future works.

2 Related Work

There are diverse threads of work on the factorization of binary matrices. The algorithmic approaches are various, ranging from continuous methods such as alternating least-squares to discrete greedy heuristics. Our first impression is that there was a divide in references around two keywords: boolean and binary.

We searched for the search keys 'Binary Matrix Factorization' and 'Boolean Matrix Factorization' using a search tool called Findpapers (GROSMAN, 2020). This search was last updated on April 26th, 2022. This tool performs a search in the bases: ACM, arXiv, bioRxiv, IEEE, PubMed, and Scopus.

We then performed an abstract screening, excluding papers that did not explicitly comment on relevant contributions to the theory, implementation, or application of BMF or its variants. During the full paper review, we also performed paper exclusion and also a brief inclusion of papers through snowballing. We made this brief inclusion because we verified common references in texts that were not included. Some related works did relevant contributions without modelling the problem as matrix factorization.

Then, we organized and reviewed the remaining references by algorithmic approaches and applications.

2.1 Algorithmic approaches

There are two central communities for decomposing the binary matrices. The first is centered around the seminal work of Zhang (ZHANG et al., 2007). Our proposed pipeline uses the algorithm proposed by the authors as a first step. This algorithm is based on alternating least squares minimization through gradient descent and some clever scaling. The other central community revolves around the discrete heuristics proposed by Miettinen in (MIETTINEN et al., 2008).

The first approach by Zhang (ZHANG et al., 2007) creates a clever algorithm that relaxes the problem such that the entries can assume any values between 0 and 1. The algorithm works by alternating gradient descent and scaling strategies for keeping entries of matrices W and H near the interval of 0 and 1 during the optimization. Thus it approximates the problem by navigating a continuous space between 0 and 1.

A problem with this approach arises when discretizing the variables by a threshold. The reconstruction error falls to a level that deems the matrices useless in practice. Later Zhang et al. explored applications for gene expression data in (ZHANG et al., 2010), which they apply to binary matrices obtained from microarray data from genetic studies. Later works, the group extend their method for symmetric binary matrices and the detecting communities and bicliques in (ZHANG; WANG; AHN, 2013) and (ZHANG; AHN, 2015). There is a practical implementation of this algorithm in the Python package called Nimfa, proposed in (ZITNIK; ZUPAN, 2012).

Also, there are constant-factor approximation algorithms with guarantees to the problem, presented by (KUMAR et al., 2019). In this paper, authors present algorithms with less error rate than Zhang's.

In (MEEDS et al., 2006) as well as in (GOLDEN; O'MALLEY, 2021), authors compare learning the dictionary matrix through many different approaches, such as sampling and clustering. The mixture matrix is non-negative, and only the dictionary matrix is binary. In contrast, in (LI et al., 2019), authors propose the use of BMF to detect changes in signals, and the mixture matrix W has binary entries, and features in the dictionary matrix H are non-negative continuous, representing the basis of signals.

Another community calls this problem Boolean Matrix Factorization and solves it through constructive heuristics. A pioneer work of (MIETTINEN et al., 2008) describes the ASSO algorithm. This algorithm uses the pairwise distance between rows to decide which position should be rounded to one, managing to maximize the coverage of the target matrix and minimize overlapping positions. It generates many potential candidates for the basis matrix H, then later tests their inclusion by calculating the gain of coverage they bring. The authors briefly present alternatives for ASSO's implementation in the same paper, including clustering and exhaustive search.

Later works in the same community of approaches include algorithm GreConD presented in (BELOHLAVEK; VYCHODIL, 2010), (BELOHLAVEK; TRNECKA, 2013), (BELOHLAVEK; OUTRATA; TRNECKA, 2014), (BELOHLAVEK; TRNECKA, 2017), algorithm Panda by (LUCCHESE; ORLANDO; PEREGO, 2013), and topFiberM in (DESOUKI; RöDER; NGOMO, 2019). All of them are based on finding good factors for the dictionary matrix H. As stated in (MIETTINEN; NEUMANN, 2021), GreConD and Panda differ in how they select factors in the first round. GreConD is based on a set cover heuristic and algorithm that tends to find non-overcovering factors. Meanwhile, Panda finds denser factors. TopFiber is a more recent algorithm that claims to be comparable in quality to GreConD and much faster (DES-

OUKI; RöDER; NGOMO, 2019). In (WAN et al., 2019), the heuristic uses row ordering and other sparse matrix methods to find dense components in the data progressively.

It is important to note that these algorithms in their respective papers are measured in their majority by a covering metric that has some problems that arise by measuring the reconstruction error using a special boolean matrix multiplication. We will discuss this in chapter 3.

In this heuristics community, the work has been extensive. One characteristic discussed in (BELOHLAVEK; OUTRATA; TRNECKA, 2018) is the quality metric for the algorithm. Authors propose a new metric to compare heuristic approaches using a linear combination of minimizing the Frobenius norm error given a rank k (DBP view) and minimizing k given a minimum desired error ϵ (AFP view). However, the reconstruction of the matrices uses the boolean matrix multiplication. As we discuss in chapter 3, this reconstruction does not take into consideration the problem of over-covering. This compensates for the greedy behavior of algorithms such as Asso, GreConD, and TopFiber.

Mirisaee et al. in (MIRISAEE; GAUSSIER; TERMIER, 2016) propose a neighborhood for searching improvements in each row. Additionally, the authors present different versions of the search by linearizing the objective function, which is an idea explored by us.

This local search deals with trading ones in different vectors that generate components. Another interesting way of thinking about local search is altering the given rank-k. In (ENE et al., 2008), they propose a local search that merges components. We didn't find any reference that has a local search for splitting components.

In recent work in (GOLDEN; O'MALLEY, 2021) and in (MALIK et al., 2021), authors propose a quantum simulated-annealing algorithm, and then they use the D-Wave quantum computer to solve this problem. In (LU et al., 2011), authors propose a Tabu search for the rank-one version of the problem. And in (SNÃ¡Å¡EL et al., 2007) and (SAENKO; KOTENKO, 2014), there are versions of a genetic algorithm, each with an interesting crossover operator.

Another idea is to solve exactly small cases using Integer Programming, such as in (KOVACS; GUNLUK; HAUSER, 2018). In the first paper, authors propose a formulation and linearize the bilinearities using McCormick envelopes (MCCORMICK, 1976). Then in (KOVACS; GUNLUK; HAUSER, 2021), the same authors suggest a column generation approach, although they test with ranks up to 6. The use of boolean multiplication linearization introduces many additional variables to the problem. In our proposed formulation, we consider a resolution heuristic of using a fixed approximation for W and solving for H and later the opposite.

This approach seems to be similar to the one taken by (SHEN; JI; YE, 2009). In this paper, the authors propose an exact formulation, a relaxation, and an approximation algorithm over the fractional result to solve the rank-one version of the problem. Other works explore this thread to solve many variants, such as weighted rank-one binary factorization in (LU et al., 2011).

Our idea of linearized alternating minimization is also explored by (HESS; MORIK; PIATKOWSKI, 2017). In this work, they propose a splitting operator procedure inspired by (PARIKH; BOYD et al., 2014).

2.2 Applications

We list a few applications commonly found in the literature for BMF algorithms. Notice that some applications carry some characteristics that call for a different formulation. For instance, recommendation and collaborative filtering are about predicting missing values. Therefore, methods have to deal with noise. Another example is general undirected networks, which have a symmetric binary adjacency matrix for factorization that imply that a single factorized matrix that represents data when squared.

2.2.1 Genetics

Some authors have proposed the application of Binary matrix factorization for microarray gene expression technology. Gene expression microarray data captures the activity of genes at the mRNA level in specific conditions. THis technology is used to understand which genes are active and their expression intensity. Analysis of this data aids in comparing gene expression profiles across conditions, identifying biomarkers indicative of certain diseases, and understanding gene functions. Standard techniques for analysis include clustering, principal component analysis, and matrix factorization.

In (ZHANG et al., 2010), authors propose using BMF to find patterns in gene expression data. In (CORRADO et al., 2014), authors also apply similar techniques of decomposing but using greedy heuristics instead and larger instances for analyzing gene expression datasets. Their datasets have tens of thousands of rows and hundreds of columns, and they group up to rank 25. We found in (EDGAR; DOMRACHEV; LASH, 2002) (CLOUGH; BARRETT, 2016) many instances similar in size to test our approach in chapter 6.

In (TU; CHEN; XU, 2011), authors proposed a BMF algorithm to detect protein complexes by clustering proteins with similar interactions through the factorization of the binary adjacency matrix of a PPI network.

Close to these kinds of instances, we also found in (BARIK; VIKALO, 2018) an application for individual haplotyping. Individual haplotyping pertains to determining sets of closely linked genetic markers on a chromosome, reflecting an individual's genetic variations. Its applications include identifying genome regions linked to diseases, studying genetic diversity within populations, and understanding drug responses based on genetics.

2.2.2 Role discovery

The seminal paper (ENE et al., 2008) presents the problem of minimizing roles in the relationship of users and access granted. A bicluster in this application maps perfectly into a system role in which a group of persons has access to the same group of functionality. There they present a heuristic to expand from a seed node into a dense bicluster, then a kind of local search to merge the discovered biclusters in this manner. Although this paper doesn't mention binary matrix factorization, it is referenced mainly by the community. A later paper (FRANK et al., 2012) does apply Boolean Matrix Factorization, but with permission request patterns from Facebook third-party applications. In (WANG et al., 2022), authors also propose a scheme for standardizing access to data from specific attributes.

Later, Saenko and Kotenko study the use of genetic algorithms applied to many virtual machine network problems for a BMF specialized in the symmetric case (SAENKO; KOTENKO, 2014), (SAENKO; KOTENKO, 2015), (SAENKO; KOTENKO, 2016) and later (PARFENOV et al., 2021). The use is interesting for finding subnetworks of dense nodes. In the same use case, Zhang has a different approach in which he proposes a projected gradient method with normalization also for the symmetric case in(ZHANG; WANG; AHN, 2013) and (ZHANG; AHN, 2015).

2.2.3 Matrix Completion, Collaborative Filtering, and Recommendation

Another application of BMF is collaborative filtering. Since the Netflix Prize (BENNETT; LANNING et al., 2007), the non-negative factorization topic has exploded with new works. A matrix of relationships between items and users can be decomposed into a small latent space of users and items that allows predicting a missing value, such as a movie that a user might like. For binary data such as "likes" or direct consumption or view of an item, we can model these matrices as boolean and apply BMF methods.

This setting has some common problems intrinsically. It is common for rows ans columns have a wide range of sparsity due to natural accumulation of consumption on some users, some items. Also, there is the challenge of working with missing data. Since viewers have not had the chance to consume all items, some are missing; the vectors of which item each user has consumed are very sparse and non-representative of a user's preferences, of what he could have consumed if he had been given a chance. So methods must be robust to noise and an immense scale for problems with many users and items.

In (RAVANBAKHSH; POCZOS; GREINER, 2015) and (RUKAT et al., 2017), and (CHEMMALAR; LAKSHMI, 2021), they discuss how to overcome the scale and the noise using different approaches for the Movie Lens dataset (HARPER; KONSTAN, 2015). As (RAVANBAKHSH; POCZOS; GREINER, 2015) uses a message passing scheme, while (RUKAT et al., 2017) uses a columns and row sampling scheme, and finally (CHEMMALAR; LAKSHMI, 2021) uses a Formal Concept discovery algorithm.

2.2.4
Outlier detection in energy signals

In (LANGE; BERGéS, 2016) and (LI et al., 2022), authors propose a different modeling of the problem in which a matrix of the decomposition is continuous and made of common patterns from various electric appliances, and the other is a binary matrix. So a factorization of this kind could detect if someone is plugging a unkown device or if an attack on the network produces outlier behavior.

2.3
The gap in the literature

We chose to position this thesis as conector between learning strategies which produce real-valued near-binary matrices, and heuristics that explore explicitly the discrete space

3
Method and proposed algorithms

Our approach combines ideas from the heuristic exact methods to post-process the results given by the gradient descent approach. We use a linearized objective function as a surrogate for the reconstruction objective error, usually calculated with the Frobenius norm. The linearized objective function allows for rewriting the problem into set notation and operations, which will be the foundation for the algorithm presented later in this section.

Before introducing the main algorithm, we first discuss some intricacies of the objective function, and a solution strategy breaks down the problem into column sub-problems. Then we show how to change the representation of the problem to operations using sets and how this relates to calculating the difference. Later we show two corollaries that drastically reduce the solution space to support a recursive search algorithm reduction.

Finally, we present the complete three-step BackDisc pipeline, in which the third step utilizes our BackColumn procedure to discretize an approximation of the factorization, recovering the loss of the discretization by thresholding.

3.1
Formulation

In our setting, the matrix $A \in \{0,1\}^{|\mathcal{M}|,|\mathcal{N}|}$ is given as input. The alternating gradient descent outputs two matrices $\tilde{W} \in [0,1]^{|\mathcal{M}|,|\mathcal{G}|}$ and $\tilde{H} \in [0,1]^{|\mathcal{G}|,|\mathcal{N}|}$, such that the multiplication $\tilde{W} \cdot \tilde{H}$ is a stable (local optimum) approximation for A. Notice that the entries of $\tilde{W}$ and $\tilde{H}$ are real values between 0 and 1. For many applications, we want to discretize these entries without losing too much of the reconstruction error, which measures the distance of approximation between A and the reconstruction $W \cdot H$.

One way to measure this approximation error is to measure how much of the variance was captured. Thus, we measure how close to zero the difference is $A - W \cdot H$. We can calculate this by measuring the norm of this difference relative to the norm of the original matrix A, as shown in equation 3-1.

$$\text{minimize } \|A - W \cdot H\|_2 = \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} (a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn}))^2 \qquad (3\text{-}1)$$

Using this equation as an optimization problem, we observe some characteristics that suggest hardness even for approximations: binary decision variables, bilinearity, and quadratic objective. Therefore, we decided to try approximating the problem using a linearized surrogate objective function using the ℓ_1-norm, as shown in equation 3-2.

$$\text{minimize } \|A - W \cdot H\|_1 = \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{N}} |a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})| \tag{3-2}$$

These linearizations allow us to rewrite the problem to solve the problem as the following mixed-integer program MIP shown below in 3-3. We introduce variable matrix B to eliminate the modulus function.

$$\begin{aligned}
\text{minimize} \quad & \mathbb{1}^T \cdot B \cdot \mathbb{1} && \text{(3-3a)} \\
\text{subject to} \quad & && \\
& A - (W \cdot H) \preceq B && \text{(3-3b)} \\
& (W \cdot H) - A \preceq B && \text{(3-3c)} \\
& && \\
& B \in \mathbb{R}_+^{|\mathcal{M}|,|\mathcal{N}|} && \text{(3-3d)} \\
& W \in \{0,1\}^{|\mathcal{M}|,|\mathcal{G}|} && \text{(3-3e)} \\
& H \in \{0,1\}^{|\mathcal{G}|,|\mathcal{N}|} && \text{(3-3f)}
\end{aligned}$$

With a linear objective function, we can use an extensive range of methods and commercial solvers to deal with the problem. Lastly, we still have to deal with the bilinearity aspect of the problem.

Inspired by other algorithms that use alternating minimization, our first investigation was to solve problems with this formulation above in an exact solver, Gurobi, as a last step after running the gradient descent. These preliminary results showed that frequently the discretization using the formulation was significantly better than any thresholding procedure.

We tried scaling the instances, but rapidly, Gurobi (Gurobi Optimization, LLC, 2023) was not able to cope with an increasing amount of binary variables. Thus, we developed an algorithm for large instances by changing the representation to a set, taking advantage of the natural sparsity of the problem.

Many matrix decomposition algorithms rely on alternating optimization. This means fixating an approximation of one of the matrices, optimizing one side, then fixating the optimized side and solving the previously fixated matrix, and repeating until achieving solution stability. We intended to apply an exact binary optimization as the last step of the alternating minimization.

Therefore, the algorithm presented can be used as a post-processing step for many approaches. In the present work, we tested it as a post-processing step for the gradient descent binary decomposition presented by Zhang et al. in (ZHANG et al., 2007) and implementation released in the package Nimfa (ZITNIK; ZUPAN, 2012).

Our approach relies on obtaining a first approximation to one of the matrices, preferably W first, and solving a subproblem problem of approximating each column of the given matrix A as a sum of columns of W, obtaining matrix H. Then, fixing the value of matrix H and solving the transposed view of the first step, approximating each row of A as a sum of a subset of rows of H, obtaining a new value for W.

3.2 The column sub-problem

Assuming a first approximation for W as fixed, a simple rearrangement of equation 3-2 shows that we can treat the summation over rows in $\mathcal{M}$ separately for each column in set $\mathcal{N}$.

$$\text{minimize} \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} |a_{mn} - (\sum_{g\in\mathcal{G}} w_{mg}\cdot h_{gn})| = \sum_{n\in\mathcal{N}} \left[\text{minimize} \sum_{m\in\mathcal{M}} |a_{mn} - (\sum_{g\in\mathcal{G}} w_{mg}\cdot h_{gn})|\right] \tag{3-4}$$

Another way of thinking about this subproblem is a problem of choosing a subset of a binary basis to represent a given binary vector. We show this interpretation in equation 3-5. In this equation, we have to approximate the column a_n using the binary decision variables h_{gn} to choose from a set of fixed basis, the columns of W.

$$\begin{bmatrix} | \\ a_n \\ | \end{bmatrix} \approxeq \begin{bmatrix} | \\ w_1 \\ | \end{bmatrix} h_{1n} + \begin{bmatrix} | \\ w_2 \\ | \end{bmatrix} h_{2n} + \cdots + \begin{bmatrix} | \\ w_g \\ | \end{bmatrix} h_{gn} + \cdots + \begin{bmatrix} | \\ w_{\mathcal{G}} \\ | \end{bmatrix} h_{\mathcal{G}n} \tag{3-5}$$

We can also run a similar procedure fixating H and optimizing matrix W, one row of A at a time, by just transposing the multiplication.

$$A_{[m\times n]} \approxeq W_{[m\times g]} \cdot H_{[g\times n]} \rightarrow A^T_{[n\times m]} \approxeq H^T_{[n\times g]} \cdot W^T_{[g\times m]} \tag{3-6}$$

Therefore, the discrete basis in this transposed view becomes a selection of rows of H to approximate each row t of matrix A.

$$\begin{bmatrix} | \\ a_m^T \\ | \end{bmatrix} \approx \begin{bmatrix} | \\ h_1^T \\ | \end{bmatrix} w_{m1} + \begin{bmatrix} | \\ h_2^T \\ | \end{bmatrix} w_{m2} + \cdots + \begin{bmatrix} | \\ h_g^T \\ | \end{bmatrix} w_{mg} + \cdots + \begin{bmatrix} | \\ h_{\mathcal{G}} \\ | \end{bmatrix} w_{m\mathcal{G}} \tag{3-7}$$

Consequently, a single algorithm for this subproblem can be used to optimally solve the linearized optimization formulation described in equation 3-4 looping through each row and then through each column.

3.3 Column Sub-Problem Example

Suppose we are trying to approximate column a_n by choosing a combination of basis and minimizing the ℓ_1-norm of the difference between the original and the reconstruction. In the example 3-8, we have the first step of this procedure. In this example, the choice of columns w_g of approximation W is decided by choosing values 0 or 1 for decision variables h_{gn}.

$$\text{minimize} \left\| \begin{bmatrix} \colorbox{blue!20}{1} \\ 0 \\ 0 \\ 0 \\ \colorbox{blue!20}{1} \\ 0 \\ \colorbox{blue!20}{1} \\ 0 \\ \colorbox{blue!20}{1} \\ \colorbox{blue!20}{1} \end{bmatrix} - \left(\begin{bmatrix} \colorbox{blue!20}{1} \\ 0 \\ \colorbox{red!20}{1} \\ 0 \\ \colorbox{blue!20}{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} h_{1n} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \colorbox{blue!20}{1} \\ 0 \\ \colorbox{blue!20}{1} \\ \colorbox{red!20}{1} \\ \colorbox{blue!20}{1} \\ \colorbox{blue!20}{1} \end{bmatrix} h_{2n} + \begin{bmatrix} 0 \\ \colorbox{red!20}{1} \\ 0 \\ \colorbox{red!20}{1} \\ 0 \\ 0 \\ \colorbox{blue!20}{1} \\ 0 \\ 0 \\ 0 \end{bmatrix} h_{3n} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \colorbox{blue!20}{1} \\ \colorbox{red!20}{1} \\ \colorbox{blue!20}{1} \\ 0 \end{bmatrix} h_{4n} \right) \right\| \tag{3-8}$$

We begin with the trivial case in which none of the basis is included in the solution. The objective function is the norm of the binary vector, which is simply the count of positions equal to one. Then we calculate the gain Δ_g of adding any column to the solution.

We calculate this gain by summing which positions in the resulting subtraction will be zero, which are the positions where both the target vector and the basis vector are one. These positions are marked with blue in equation 3-8. Then we subtract the superfluous positions where the basis vectors have ones, and the target vector have not. Subtracting a column with an extra one in the target vector will increase its modulus. Those positions are marked with light red in equation 3-8.

$$\begin{aligned} OF = 5; \qquad h_n &= [0;0;0;0]; && \text{(3-9a)} \\ \Delta_{1n} &= 2 - 1 = 1; && \text{(3-9b)} \\ \Delta_{2n} &= 4 - 1 = 3; && \text{(3-9c)} \\ \Delta_{3n} &= 1 - 2 = -1; && \text{(3-9d)} \\ \Delta_{4n} &= 2 - 1 = 1 && \text{(3-9e)} \end{aligned}$$

Then we calculate for each variable h_{gn} in $\mathcal{G}$ a gain indicator Δ_{gn}. This gain is calculated by subtracting the number of positions in which the target vector a_n has in common with the basis by the superfluous. For the next step, we add h_{2n} to the solution. In equation 3-10, we will recalculate the target vector by subtracting the added basis and repeat the procedure.

$$\text{minimize} \left| \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{bmatrix} - \left(\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} h_{1n} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} h_{3n} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} h_{4n} \right) \right| \tag{3-10}$$

In this step, we observe that the target vector now carries an entry equal to minus one. Even if a new basis has a one in this position, the ℓ_1 norm of the target vector will increase because the modulus in this position will also increase. Therefore, the only case in which there is an actual gain is to cover the remaining positions in which the target vector is equal to one.

$$\begin{aligned} OF = 2; \qquad h_n &= [0;1;0;0]; && \text{(3-11a)} \\ \Delta_{1n} &= 1 - 2 = -1 && \text{(3-11b)} \\ \Delta_{3n} &= 0 - 3 = -3 && \text{(3-11c)} \\ \Delta_{4n} &= 0 - 3 = -3 && \text{(3-11d)} \end{aligned}$$

3.3

Also, notice that by adding a basis to the solution, the number of remaining positions left with value one can only decrease monotonically. When we add a column to a solution tentatively, the number of ones not covered only reduces. Thus, the Δ_{gn} of the remaining columns can only monotonically decrease. This monotonicity has some interesting consequences.

Firstly, when we find a case in which all remaining Δ is negative, it is a local optimum because Δ's which are negative will never become positive later by progressively adding basis to the solution set.

Additionally, we didn't even need to consider h_{3v} with a respective negative Δ_{3n} from equations 3-9 for the second round. Since Δ_{3n} in step one depicted in equations 3-8 was negative, it can only decrease. Consequently, it does not need to be considered in further steps.

Also, summing positive deltas is an upper bound for any possible combination of adding g to the solution set. For instance, the sum of Δ_{2n} and Δ_{4n} in the first step is two, as shown in equations 3-9. Consequently, when a recursive algorithm finds a local optimum with objective 2 to the first state with objective 5, we know that the best we can achieve by exploring indices 2 and 4 is an objective of value 3. Thus we can stop searching and prove that the solution found in step two, as shown by equations , is optimal.

3.4 The Set representation

For dealing with really large matrices, we can take advantage of the structure of the problem transforming the many binary vectors into sets that contain the positions in which these vectors are equal to one.

Let $\mathcal{I}$ a function that transforms vectors to a set of positions equal to one. We can apply this function to any rows or columns of matrices A, W, or H during the column subproblem solve.

Definition 3.1 (Function $\mathcal{I}$ that translates sparse vectors to sets)
Let $x \in \{0,1\}^n$. Then the function $\mathcal{I}(x) : \{0,1\}^n \to 2^n$ is the set of indices of the positions of x which are greater or equal to 1.

Examples of usage of the function $\mathcal{I}$. Let $x = [0,1,1]$. Then $\mathcal{I}(x) = \{2,3\}$. Or let $y = [1,1,0]$. Then $\mathcal{I}(y) = \{1,2\}$.

An algorithm for the column subproblem using a fixed W has to decide which positions of vector h_n should be explored, fixating to 1. The algorithm starts with the trivial solution such that all positions are assigned to zero. The vector h_n has size $|\mathcal{G}|$. Therefore, using the set representation, we begin the algorithm with:

$$\mathcal{I}(h_n) = \varnothing$$

since all positions initially are zero.

We add $g \in \mathcal{G}$ to $\mathcal{I}(h_n)$ whenever we are investigating assigning position g in vector h_n to 1. As the algorithm takes steps t it inserts into the current solution some basis g from $\mathcal{G}$

$$\mathcal{I}(h_n) \subset \mathcal{G}$$

Definition 3.2 (Set $\mathcal{Q}$ of remaining positions to cover in the target vector) *The algorithm keeps track of the set $\mathcal{Q}$ of uncovered positions of the target vector a_n. The set $\mathcal{Q}$ is a subset of $\mathcal{M}$.*

$$\mathcal{Q} = \mathcal{I}(a_n) \setminus \left(\bigcup_{g}^{\mathcal{I}(h_n)} \mathcal{I}(w_g) \right)$$

If a new g is added to solution set $\mathcal{I}(h_n)$ then we can update $\mathcal{Q}$ in the following manner:

$$\mathcal{Q} := \mathcal{Q} \setminus \mathcal{I}(w_g)$$

Lemma 3.3 (Gain calculation using set representation) *When solving for a_n, the increment Δ_{gn} of adding g to a solution $\mathcal{I}(h_n)$ is calculated as:*

$$\Delta_{gn} = |\mathcal{I}(w_g) \cap \mathcal{Q}| - |\mathcal{I}(w_g) \setminus \mathcal{Q}|$$

Proof.

The set $\mathcal{Q} \subset \mathcal{M}$ of uncovered positions in a_n partitions the set $\mathcal{I}(w_g)$ into two. Firstly, the intersection between $\mathcal{I}(w_g)$ and $\mathcal{Q}$ represent the uncovered positions of a_n which could be covered by adding g to the solution $\mathcal{I}(h_n)$. This would contribute positively to minimizing the objective function.

The remaining elements of $\mathcal{I}(w_g)$ which are not in $\mathcal{Q}$ would contribute negatively because they are either superfluous (do not cover any positions in a_n) or they were already covered by another group w_g during the construction of a solution.

We will give a direct proof. Suppose we add g' to the solution set $\mathcal{I}(h_n)$. Let g' as a fixed position in h_n we wish to flip to one and let $\Delta_{g'n}$ the difference in objective when changing the value of $h_{g'n}$ from zero to one. The objective function for solving just the column a_n is:

$$\text{minimize} \sum_{m \in \mathcal{M}} |a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})|$$

First, we remember that all elements a_{mn}, w_{mg}, h_{gn} are either $\{0,1\}$ by definition of the problem. So, for each g the multiplication $w_{mg} \cdot h_{gn} \in \{0,1\}$ also. Additionally, $0 \leq \sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn} \leq |\mathcal{G}|$.

The summation $\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn}$ also can be simplified to $\sum_{g \in \mathcal{I}(h_n)} w_{mg} \cdot h_{gn}$ for a given solution $\mathcal{I}(h_n)$, because if $g \notin \mathcal{I}(h_n)$ then $h_{gn} = 0$.

The term $a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})$ can assume values from 1 to $-|\mathcal{G}|$.

$a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})$	1,	0,	-1,	-2,	$\dots$,	$-\lvert\mathcal{G}\rvert$
$\lvert a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})\rvert$	1,	0,	1,	2,	$\dots$,	$\lvert\mathcal{G}\rvert$

So, when $h_{g'n}$ becomes one, for all t in which $w_{mg'}$ is one will have the summation $(\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})$ increase also one in value. Which means go right on the above scale. So, the only possibility for minimizing the objective function is to use the summation to cover a position where $a_{mn} = 1$. Then the module will decrease from 1 to 0. Otherwise, the module will only increase.

For each position $t \in \mathcal{I}(w_{g'})$, the change in the value of the term $|a_{mn} - (\sum_{g \in \mathcal{G}} w_{mg} \cdot h_{gn})|$ will fall into 3 cases:

Case 1: $a_{m'n} = 0$
only the summation over $\mathcal{G}$ increases, so the objective function also increases.

Case 2: $a_{m'n} = 1$ and $(\sum_{g \in \mathcal{I}(h_n)} w_{mg} \cdot h_{gn}) > 0$ in this case, the summation has enough magnitude to cancel out $a_{mn} = 1$, so the increase in the summation increases the objective function

Case 3: $a_{m'n} = 1$ and $(\sum_{g \in \mathcal{I}(h_n)} w_{mg} \cdot h_{gn}) = 0$. Only in this case, when the summation is equal to 0 in that position that the absolute value decreases because it will *cover* the position a_{mn}. Therefore, the only thing that we must track is the uncovered positions a_{mn}. We will do it maintaining a set $\mathcal{Q} \subset \mathcal{I}(a_n) \subset \mathcal{M}$ in which we deduce the positions t in which $a_{mn} = 1$ and the summation over $\mathcal{G}$ is still equal to 0, this means, is not covered. ■

With this lemma, we can now prove the main foundation for this work. The following theorem will allow for an efficient search in practice.

Theorem 3.4 (Contribution decreasing monotonicity) *Whenever adding g to solution $\mathcal{I}(h_n)$, all the gains of adding any other element in the solution in the next steps can only stay the same or decrease. This means when recalculating all other $\Delta_{g'n}$ of g' not yet in the solution set $\mathcal{I}(h_n)$, the new value is lesser or equal than it was before.*

Proof. By Lemma 1, we have that $\Delta_{gn} = |\mathcal{I}(w_g) \cap \mathcal{Q}| - |\mathcal{I}(w_g) \setminus \mathcal{Q}|$.
When you add $\{g\}$ to the solution set $\mathcal{I}(h_n)$, we update the set $\mathcal{Q}$ by subtracting the newly covered positions. So, for the remaining positions g', the $\Delta_{g'n}$ is updating taking into consideration that $\mathcal{Q} := \mathcal{Q} \setminus \mathcal{I}(w_g)$. Therefore,

$\mathcal{Q}$ has fewer items than before, and $|\mathcal{I}(w_g) \cap \mathcal{Q}|$ becomes less or equal than before and $|\mathcal{I}(w_g) - \mathcal{Q}|$ becomes greater or equal than before. Consequently, the value of Δ_{gn} can only decrease or stay the same. ■

Theorem 3.4 has many interesting consequences. The monotonicity can be used to define local optima and to eliminate positions to search during a recursive enumeration. This enables the design of a backtracking algorithm that finds the global optimum for the sub-problem and only explores a small subset of the combinatorial decision space.

Corollary 3.5 (Negative contributing candidates skipping) *If $\Delta_{g'n}$ associated with any remaining $g' \notin \mathcal{I}(h_n)$ is negative, then g' will never be in any local optima solution with the g that belong to the current solution set $\mathcal{I}(h_n)$.*

Proof. Since the $\Delta_{g''n}$ of every position g'' only decreases when adding any other g' to the solutions set $\mathcal{I}(h_n)$, then adding g' with negative $\Delta_{g'n}$ would not only leave the objective function worse, but it would also worsen all the other $\Delta_{g''n}$, the potential of constructing better solutions.

So, there always exists a solution better than one constructed by adding g', a solution that simply skipped adding g' will stay ahead. Therefore, a solution containing g' could not be a local optimum, nor a global optimum consequently. ■

Corollary 3.6 (Early stopping upper-bound) *The solution set $\mathcal{I}(h_n)$ is a subset of $\mathcal{G}$. Any subset $\mathcal{P}$ of $\mathcal{G}$ disjoint from $\mathcal{I}(h_n)$ can have its overall upper-bound calculated as.*

$$\Delta_{UB} = \sum_{g \in \mathcal{P}} max(\Delta_{gn}, 0)$$

This means if any subset of $\mathcal{P}$ is added to the solution set $\mathcal{I}(h_n)$, the overall contribution to the objective function is bounded by Δ_{UB}.

Proof. The *max* function is just a mechanism to select the positive Δ_{gn}. So suppose that any $g \in \mathcal{P}$ is added to solution set $\mathcal{I}(h_n)$ then by theorem 1, all the remaining Δ_{gn} are updated to be of a lesser or equal value. Then, the sum of all the positive $\Delta_g v$ for all g before adding is greater than the actual Δ_{gn} at the point of adding them to the solution and updating the objective function.

Thus, Δ_{UB} is greater than the overall gain of adding any subset of $\mathcal{P}$ in any order. Consequently, if $\mathcal{P}$ is the set of remaining candidate g to explore, and we know that exists a solution OF^* lesser than the current one $OF - \Delta_{UB}$, we don't need to explore $\mathcal{P}$. Because any solution would be worse than the one with value OF^*. ■

If we assume a sequential inclusion of candidate bases to the solution at any given point during the search, we can sum positive deltas remaining to explore and calculate an upper bound of the contribution of any combination of insertions of the associated bases. This means that if we already know any solution, this fact can be used to prove that we don't need to further explore a significant part of the decision space.

3.5 Set Notation Nomenclature

The set notation introduces sparsely many new symbols to the discussion. In this section, we review them briefly before revisiting the example using the new notation and introducing the more general algorithm for solving the column subproblem.

$\mathcal{M}$, $\mathcal{N}$, $\mathcal{G}$	are the sets of all rows, columns, and principal components (groups), respectively.
$A_{[\lvert\mathcal{M}\rvert \times \lvert\mathcal{N}\rvert]}$	is the given matrix we are going to factorize it has $\lvert\mathcal{M}\rvert$ rows and $\lvert\mathcal{N}\rvert$ columns.
a_m or a_n	are a short notation for row $m \in \mathcal{M}$ of matrix A or column $n \in \mathcal{N}$ of matrix A .
$W_{[\lvert\mathcal{M}\rvert \times \lvert\mathcal{G}\rvert]}$	is the binary mixture matrix with $\lvert\mathcal{M}\rvert$ rows and $\lvert\mathcal{G}\rvert$ columns.
w_g	is a short notation for column g of matrix W
$H_{[\lvert\mathcal{G}\rvert \times \lvert\mathcal{N}\rvert]}$	is the binary dictionary or factor matrix W with $\lvert\mathcal{M}\rvert$ rows and $\lvert\mathcal{G}\rvert$ columns.
$\tilde{W}, \tilde{H}$	we utilize the $\tilde{\cdot}$ notation to denote matrices with near-binary real-valued entries
$\overline{W}, \overline{H}$	we utilize the $\bar{\cdot}$ notation to denote a greedily discretized (usually low-quality) approximation for binary matrices

$\mathcal{I}(\cdot)$	is the function that transforms a binary vector into the set of the positions in it which are equal to one.
$\mathcal{I}(a_n)$	is the set of positions of column n of matrix A that are equal to one.
$\mathcal{I}(h_n)$	is the solution set containing the attempted positions in column n which we will assign to one, once we reach a final solution for column h_n
$\mathcal{I}(w_g)$	is the set of positions of column g of matrix W which are equal to one, in the context of the algorithm, is the basis we will decide if is worth for covering uncovered positions in column a_n.
$\mathcal{Q}$	is the set of uncovered positions of the target row (or column); it begins as $\mathcal{I}(a_n)$ and progressively, as the algorithm adds base g, it deduces $\mathcal{I}(w_g)$ from it.
OF	is the current objective function obtained using the norm 1 as a surrogate.
PQ	is a generic priority queue which removes groups g with the greatest Δ_{gn}
Δ_{gn}	is the contribution in the objective function of adding g to solution set $\mathcal{I}(h_n)$ or flipping h_{gn} from zero to one.
Δ_{UB}	is an upper bound obtained from summing the remaining Δ_{gn} in queue PQ
$\mathcal{I}(h_n)_{rec}$, OF_{rec}	are the best solution and objective value found returning from a child recursion
$\mathcal{I}(h_n)_*$, OF_*	are trackers of the best solution and objective values found in the loop of recursions

3.6 The example revisited

With the theoretical basis, we can revisit the first example using the set representation. Given the matrix A, we obtain, using any means, an approximation for W, which we treat as fixed for the column subproblem. For a target column a_n, we search for the best combinations of the bases w_1 through w_4 that, when summed, are the best approximation for it. We apply the function $\mathcal{I}$ to each of the columns of W and target vector a_n, which is a column of A.

$$\begin{aligned}
\mathcal{I}(a_n) &= \{1, 5, 7, 9, 10\} && \text{(3-12a)}\\
\mathcal{I}(w_1) &= \{1, 3, 5\} && \text{(3-12b)}\\
\mathcal{I}(w_2) &= \{5, 7, 8, 9, 10\} && \text{(3-12c)}\\
\mathcal{I}(w_3) &= \{2, 4, 7\} && \text{(3-12d)}\\
\mathcal{I}(w_4) &= \{7, 8, 9\} && \text{(3-12e)}\\
& && \text{(3-12f)}
\end{aligned}$$

Then we calculate the gain of adding each of the bases to the solution using Lemma 3.3.

$$\begin{aligned}
\mathcal{I}(h_n) &:= \varnothing && \text{(3-13a)}\\
\mathcal{Q} &:= \mathcal{I}(a_n) = \{1, 5, 7, 9, 10\} && \text{(3-13b)}\\
OF &:= \|\mathcal{I}(a_n)\| = 5 && \text{(3-13c)}\\
\Delta_{1n} &= \|\mathcal{I}(w_1) \cap \mathcal{Q}\| - \|\mathcal{I}(w_1) - \mathcal{Q}\| = \|\{1, 5\}\| - \|\{3\}\| = 1 && \text{(3-13d)}\\
\Delta_{2n} &= \|\mathcal{I}(w_2) \cap \mathcal{Q}\| - \|\mathcal{I}(w_2) - \mathcal{Q}\| = \|\{5, 7, 9, 10\}\| - \|\{8\}\| = 3 && \text{(3-13e)}\\
\Delta_{3n} &= \|\mathcal{I}(w_3) \cap \mathcal{Q}\| - \|\mathcal{I}(w_3) - \mathcal{Q}\| = \|\{7\}\| - \|\{2, 4\}\| = -1 && \text{(3-13f)}\\
\Delta_{4n} &= \|\mathcal{I}(w_2) \cap \mathcal{Q}\| - \|\mathcal{I}(w_2) - \mathcal{Q}\| = \|\{7, 9\}\| - \|\{8\}\| = 1 && \text{(3-13g)}\\
& && \text{(3-13h)}
\end{aligned}$$

Observe in equations 3-13 that we already have position 3 with a negative delta. Since the contributions are monotonically decreasing, we do not need to consider it again in further steps.

$$\text{Second call} \tag{3-14a}$$
$$\mathcal{I}(h_n) := \mathcal{I}(h_n) \cup \{2\} = \{2\} \tag{3-14b}$$
$$\mathcal{Q} := \mathcal{Q} - \mathcal{I}(w_2) = \{1\} \tag{3-14c}$$
$$OF := OF - \Delta_{2n} = 5 - 3 = 2 \tag{3-14d}$$
$$\Delta_{1n} := \|\mathcal{I}(w_1) \cap \mathcal{Q}\| - \|\mathcal{I}(w_1) - \mathcal{Q}\| = \|\{1\}\| - \|\{3,5\}\| = -1 \tag{3-14e}$$
$$\Delta_{4n} := \|\mathcal{I}(w_2) \cap \mathcal{Q}\| - \|\mathcal{I}(w_2) - \mathcal{Q}\| = \|\varnothing\| - \|\{7,8,9\}\| = -3 \tag{3-14f}$$
$$\tag{3-14g}$$

In equations 3-14, we added base 1 to the solution and eliminated base 3 because it had a negative contribution. The remaining bases, 1 and 4, now have negative contributions. Therefore, there is no way of further adding any basis to the solution without worsening the objective function. In a recursive scheme, the procedure should go back to the state described by equations 3-13. Now the differences are that we already explored adding 2 to the solution set $\mathcal{I}(h_n)$ and that there is a local optimum with an objective function equal to 2.

The natural approach is to choose between bases 1 and 4 to begin a new search. However, both their respective Δ's, when summed up, are equal to 2, which we know is an upper bound for the contribution of adding them in any combination in any order. Since the current objective functions are back to 5, and the contribution upper-bound is 2, the best we can expect by adding these bases is 3, which is more than the solution we already found with just the basis 2. Therefore, we also do not need to search using bases 1 and 4. Finally, since there are no options to explore, we can conclude that the local optimum we found was actually the global optimum.

3.7 The BackColumn algorithm for the column sub-problem

We have introduced sufficient elements to design a recursive procedure that efficiently explores the decision space of the column subproblem. The main ideas are to calculate the individual contributions of all bases, consider only the ones with positive contributions, and begin adding the ones with the most significant contribution. Later in this chapter we present how to iterate and apply this algortihm over all columns and rows to obtain the binary matrix factorization discretization.

In a recursive scheme, our idea is to design a procedure that only controls one inclusion step. In further recursive calls, the Δ's tend to rapidly diminish to negative values, and we can loop through the decision space efficiently. We provide a pseudo-code for this procedure in algorithm 1.

Algorithm 1: Our proposed BackColumn algorithm for the column sub-problem

```
function BackColumn($W$, $\mathcal{G}$, $\mathcal{Q}$, $\mathcal{I}(h_n)$, $OF$):
    Let $PQ$ := empty priority queue
    Let $\mathcal{I}(h_n)^* := \mathcal{I}(h_n)$
    Let $OF^* := OF$
    for g in ($\mathcal{G} - \mathcal{I}(h_n)$):
        $\Delta_{gn} = |\mathcal{I}(w_g) \cap \mathcal{Q}| - |\mathcal{I}(w_g) - \mathcal{Q}|$ // From Theorem 1
        if $\Delta_{gn} > 0$: // From Corollary 1
            add $g$ with priority $\Delta_{gn}$ to $PQ$
    while PQ is not empty queue:
        Let $g$, $\Delta_{gn}$ := remove first priority item from $PQ$
        $\mathcal{I}(h_n)_{rec}$, $OF_{rec}$ := BackColumn($W$,
                                             $\mathcal{G} := \{g | g \in PQ\}$,
                                             $\mathcal{Q} := \mathcal{Q} - \mathcal{I}(w_g)$,
                                             $\mathcal{I}(h_n) := \mathcal{I}(h_n) \cup \{g\}$,
                                             $OF := OF - \Delta_{gn}$)
        if $OF_{rec} < OF^*$:
            $OF^* := OF_{rec}$; $\mathcal{I}(h_n)^* := \mathcal{I}(h_n)_{rec}$
        $\Delta_{UB} := \sum_{g \in PQ} \Delta_{gn}$
        if $OF - OF^* > \Delta_{UB}$ break // From Corollary 2
    return $\mathcal{I}(h_n)^*, OF^*$
BackColumn($W$, $\mathcal{G}$:=$\mathcal{G}$, $\mathcal{Q} := \mathcal{I}(a_n)$, $\mathcal{I}(h_n) := \varnothing$, $OF := |\mathcal{I}(a_n)|$)
```

The algorithm's inputs are the fixed matrix W and four other data structures that describe the state of an exhaustive search. The state of the search can be controlled using three sets and one integer. Internally the algorithm keeps a priority queue PQ of candidate positions to include in the solution set. In a loop, the algorithm makes a recursive call to itself, updating the data structures which represent the subproblem of searching once a candidate basis g is added to solution set $\mathcal{I}(h_n)$.

The set $\mathcal{G}$ represents remaining positions to explore. At the root of the recursive calls, we assign $\mathcal{G}$ to the set of all possible indexes we can add to the solution set $\mathcal{I}(h_n)$. At the beginning of the step, we filter indexes $g \in \mathcal{G}$ with positive contributions $\Delta_{gn} > 0$ to a priority-queue PQ. The algorithm loops through all positions greedily, choosing the most significant contribution in the priority queue and then calling recursively the same procedure. The recursive call narrows the space of remaining candidates to those remaining in the priority queue, that is, those with positive contributions that are not yet explored in the loop. Thus we assign to $\mathcal{G}$ in the recursive call just the elements in PQ.

The set $\mathcal{Q}$ represents the uncovered positions of the original target vector a_n. The set $\mathcal{Q}$ is initialized as the original positions in a_n, which are equal to one. This set allows calculating the contributions Δ_{gn} efficiently. In the recursive call, when we add g to the solution set $\mathcal{I}(h_n)$, we must update what are the positions left to cover by subtracting the set $\beth_{\}}$ of positions covered this basis.

Also, we have the current objective function represented as a floating-point variable OF. In the recursive call, we simply update the OF by subtracting the contribution Δ_{gn} associated with the candidate g added to the solution set.

Finally, the algorithm will keep track of the best solution found in each recursion made in each loop. If the solution $\mathcal{I}(h_n)_{rec}$ found in the recursion is better than the best solution $\mathcal{I}(h_n)^*$ found previously, the algorithm will store its value.

3.8 The intuition behind the cut-off gains

The column solve algorithm works as an unconventional recursive branch-and-bound procedure. At the top level, there is a while loop that controls the addition of candidate columns g to the solution set $\mathcal{I}(h_n)$. Each time the algorithm adds one candidate to the solution set, it calls itself recursively, updating the state of the problem in terms of the positions that are still uncovered $\mathcal{Q}$ and the remaining valid column candidates $\mathcal{G}$.

We propose a visualization of this recursive call in figure 3.1. The rectangles with rounded corners represent each call to the BackColumn procedure. In their top-right corner, we represent the current state's solution set $\mathcal{I}(h_n)$. For each candidate column in the priority queue, there will be a recursive call that adds it to the solution and updates the state of the problem. The remaining candidates not added to the solution are passed as arguments in the recursive call. Recursive calls are represented as blue arrows, and the while loop iterations through priority queue PQ are represented as green arrows.

In this example, we first follow the blue arrow, a recursive call that adds column 1 to the solution set $\mathcal{I}(h_n)$. The remaining candidates $\{2, 3\}$ are passed as arguments for the child procedure as remaining candidates to be analyzed. Then this procedure calls itself recursively adding 2 to the solution set, then another call adds 3 to the solution. The new call has an empty priority queue; then it defaults to returning the solution. Consequently, the recursive

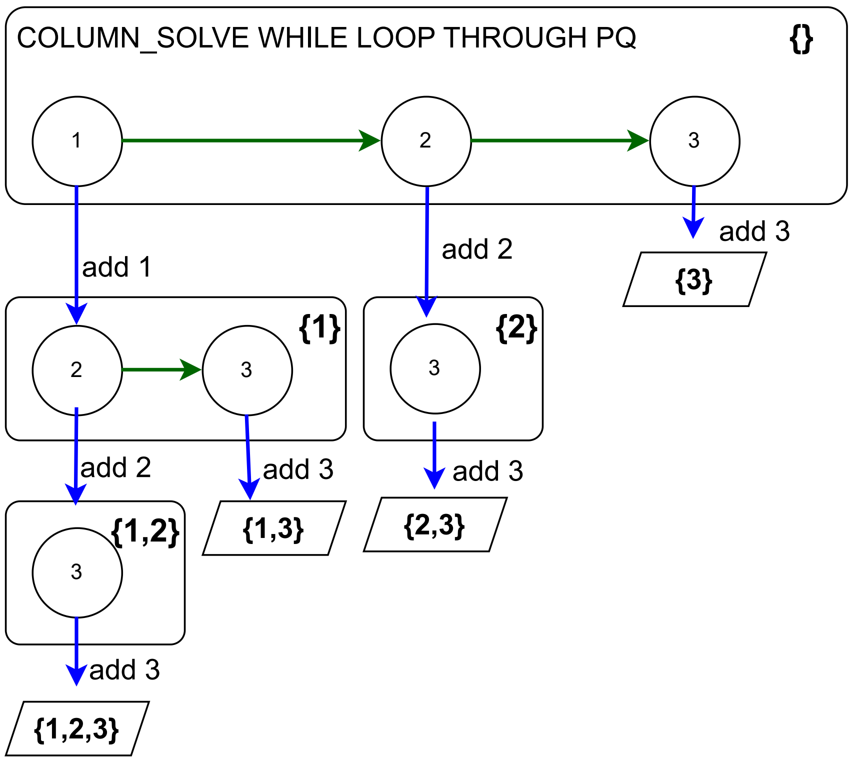


Figure 3.1: Example recursion tree of the main BackColumn procedure.

call compares the solution $\{1, 2, 3\}$ to the solution $\{1, 2\}$ keep the best and returns. Then, we explore the first green arrow, adding the candidate 3 to the solution set, obtaining $\{1, 3\}$. Thus, when the green arrow is used in practice, it explores a solution space without the candidates it leaves behind.

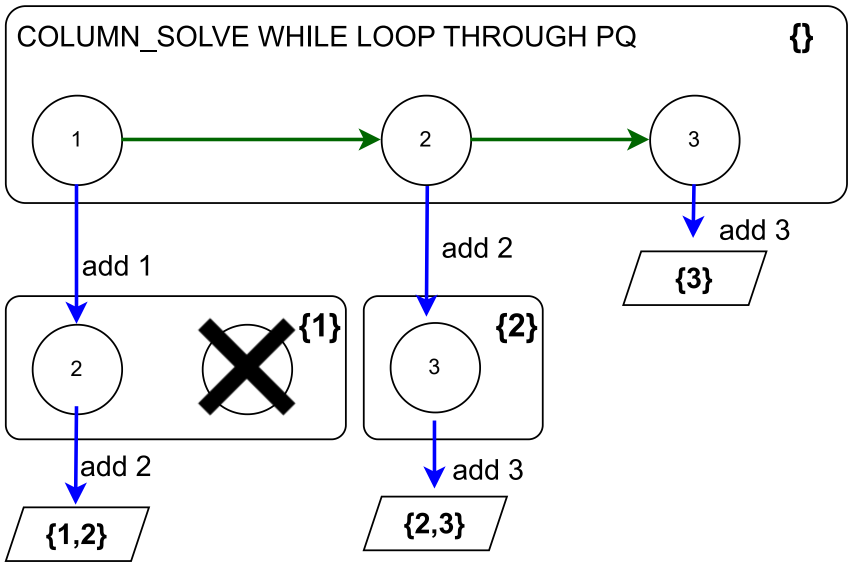


Figure 3.2: Example recursion tree revisited using the corollary 3.5.

In the hypothetical situation presented in figure 3.2, Δ_3 got negative after adding the solution candidate column 1 to solution set $\mathcal{I}(h_n)$, so we can eliminate candidate 3 and it is not passed in the next recursive call. Therefore, we avoid the unnecessary exploration of solutions $\{1, 2, 3\}$ and $\{1, 3\}$. Notice that this corollary also allows us to check for local optima on the leaves. Because whenever there is still a positive delta to be explored in the priority queue, there is probably a better solution to be found by recursing.

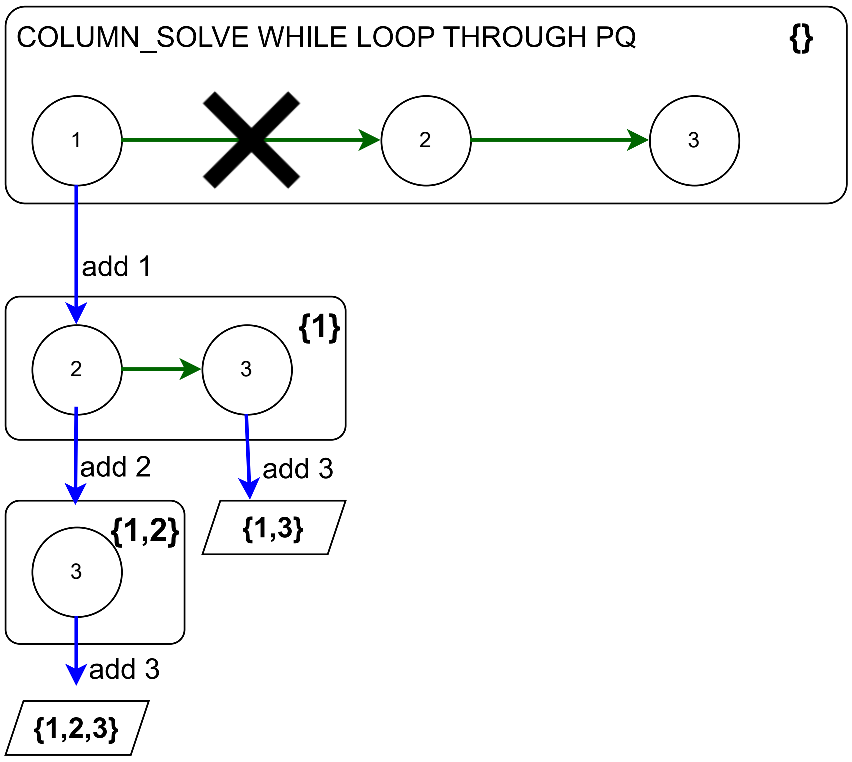


Figure 3.3: Example recursion tree revisited using the corollary 3.6.

In the hypothetical situation presented in figure 3.3, after recursing through the inclusion of 1 at the top of the recursive call pile, $\Delta_2 + \Delta_3$ were found to be less or equal to gain already obtained from a known solution. Thus the whole while loop can be halted. In this situation, by stopping early, we can avoid the exploration of all combinations of inclusions of remaining candidates.

3.9 Proposed pipeline and the BackDisc Algorithm

Our final approach for factoring binary matrices is to generate a near binary factoring using a three-step pipeline, as shown in 5.1. First, the gradient descent algorithm described in (ZHANG et al., 2007) and implemented by (ZITNIK; ZUPAN, 2012) producing $\tilde{W}$ and $\tilde{H}$. Then, using a simple search for thresholds to discretize $\overline{W}$ then calling the BackColumn procedure for every column of A, then for every row. We describe both the simple threshold search procedure and the discretization procedure in this section.

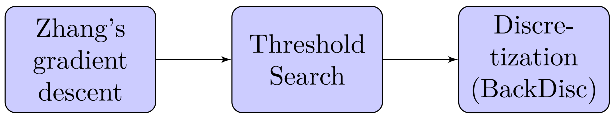


Figure 3.4: The BackDisc pipeline

3.9.1 Thresholding

The thresholding procedure 2 searches exhaustively for the pair of real-valued thresholds $t_w, t_h \in \{0, 1\}$ that minimize the Frobenius Relative Error.

Algorithm 2: The Thresholding search algorithm, the second step of the BackDisc pipeline.

```
function threshold_search($A$, $\tilde{W}$, $\tilde{H}$):
    $\text{norm}_{\text{best}} := \infty$
    for $t_w$ in $[.05, .10, .15, ..., .85, .90, .95]$:
        $\overline{W} := \text{threshold}(\tilde{W}, t_w)$
        for $t_h$ in $[.05, .10, .15, ..., .85, .90, .95]$:
            $\overline{H} := \text{threshold}(\tilde{H}, t_h)$
            $\text{norm} := \left\|A - \overline{W} \cdot \overline{H}\right\|_2 / \|A\|_2$
            if $\text{norm} < \text{norm}_{\text{best}}$ :
                $\text{norm}_{\text{best}} := \text{norm}$
                $\overline{W}_{\text{best}} = \overline{W}$
                $\overline{H}_{\text{best}} = \overline{H}$
    return $\overline{W}_{\text{best}}$, $\overline{H}_{\text{best}}$
```

For matrices $\tilde{W}, \tilde{H}$, we try every combination for these thresholds that are multiples of 5% and fixate to one entry above the threshold and fixate to zero otherwise. This procedure generates two greedily discretized matrices $\overline{W}, \overline{H}$. We test the reconstruction error for this pair of matrices and store the matrices that minimize the error.

One of the motivations for this work is that even though we search for these matrices, the Frobenius error is still inferior to that obtained by Zhang's algorithm. In practice, the discretization procedure we propose in the following chapter serves to recover the loss by thresholding.

3.9.2 BackDisc Algorithm

As we showed in equations 3-5 and 3-7, the linearized version of the problem, in addition to fixating some of the matrices, can be algebraically divided into solving a column subproblem for each of the columns of the original matrices and, in the transposed view, each of the rows of the original matrix.

Our main contribution is using a new scalable approach for solving the column sub-problems to post-process matrices factorized by gradient-descent approaches. Gradient-based methods can deal with large matrices and provide the first approximation needed to solve these problems. Internally, they use alternating minimization until they converge in two almost binary matrices with low reconstruction error. As we discussed before, other approaches in related work can either: (1) obtain matrices with almost binary entries that, when are thresholded, have a significant loss in precision or (2) obtain matrices with truly binary entries but with small sizes.

Our first intuition was to take these matrices and solve exactly the linearized formulation over each matrix W and H. Thus we loop through the rows of a given matrix A and the approximations yielded by gradient descent algorithms W and H. We fixate W and use its columns as a basis to recalculate each row of H by looping through the columns of A. Then we run a similar procedure for the transposed view.

We show the pseudo-code for this procedure in 3.

Algorithm 3: Our proposed BackDisc algorithm that uses the BackColumn algorithm for recovering the thresholding error loss.

```
function BackDisc($A$, $\overline{W}$):
    Let $|\mathcal{M}|, |\mathcal{N}|$ := dimensions of target matrix $A$
    Let $|\mathcal{M}|, |\mathcal{G}|$ := dimensions of approximation matrix $\overline{W}$
    Let $H$ := matrix of zeros with dimensions $|\mathcal{G}|$, $|\mathcal{N}|$
    for $v$ in $\mathcal{N}$:
        $\mathcal{I}(h_n)^*, OF^*$ := BackColumn($a_n$, $\overline{W}$, $\mathcal{G}$, $\mathcal{Q} := \mathcal{I}(a_n)$, $\mathcal{I}(h_n) := \varnothing$, $OF := |\mathcal{I}(a_n)|$)
        $h_n$ := reconstruct column $h_n$ of $H$ assigning values 1 to positions $\mathcal{I}(h_n)^*$
    Let $W$ := matrix of zeros with dimensions $|\mathcal{M}|$, $|\mathcal{G}|$
    for $t$ in $\mathcal{M}$:
        $\mathcal{I}(w_m)^*, OF^*$ := BackColumn($a_m$, $H$, $\mathcal{G}$, $\mathcal{Q} := \mathcal{I}(a_m)$, $\mathcal{I}(w_m) := \varnothing$, $OF := |\mathcal{I}(a_m)|$)
        $w_m$ := reconstruct rows $w_m$ of $W$ assigning values 1 to positions $\mathcal{I}(w_m)^*$
    return $W$, $H$
```

At a glance, we are fixating a matrix W and solving the linearized problem for each row of H. Then we fixate H and solve the linearized problem for each row of W just once. We solve these two problems by separating each column of A and running the BackColumn procedure in a loop. Then we apply the same method to the transposed view of the problem by fixating the rows of H calculated previously and solving the linearized problem for each row of transposed W.

3.10 The PCA/SVD lowerbound

There is a well-established fact that the Truncated Singular Value Decomposition or Principal Component Analysis produces the best approximation of rank-k to any given real-valued matrix (ECKART; YOUNG, 1936) and (GOLUB; LOAN, 2012).

So, let's:

$$A \xrightarrow[\text{k-SVD}]{\text{PCA}} U_k \cdot \Sigma_k \cdot V_k^T$$

the truncated SVD decomposition of A, in which U_k and V_k are orthonormal rotation matrices with k columns and Σ is a diagonal square matrix with k singular values, that work as a directional stretchers.

So, the error is calculated by:

$$\left\| A - U_k \cdot \Sigma_k \cdot V_k^T \right\|_2 / \left\| A \right\|_2$$

is the deterministic minimum possible error that any decomposition could have using rank-k for any real-valued or binary-valued given matrix A.

Therefore, in all the applications, we use this error as a lower bound and a measure of the hardness or informally the suitability for grouping the instance. That means some instances are so hard to represent in lower ranks that the error is still high even in a more general factorization with the guaranteed best approximation.

We expect any approach to factoring the matrix into binary matrices to produce a result with a much higher Frobenius norm error. However, the lower bound provided by the PCA allows us to understand if the problem instance is intrinsically hard for grouping or factoring.

3.11 The problem of the Boolean Matrix Multiplication

Many previous works measure the loss reconstructing the A using an operation they call boolean matrix multiplication and a covering indicator. We chose not to use such an approach for two reasons.

First, it disregards the problem of overcovering a position with more than one component. Thus, it is a more forgiving error metric. Also, and most importantly, it makes sense only when comparing strictly boolean (binary) results, whereas our approach includes intermediate steps that are real-valued. It would be impossible to compare if there was a loss or a gain against real-valued results.

Definition 3.7 (Boolean matrix multiplication (BoolMM)) *Let binary/boolean matrices $W \in \{0,1\}^{m\times k}$ and $H \in \{0,1\}^{m\times k}$. Their boolean matrix multiplication (BoolMM) aggregates the k components by OR ($\vee$) operation instead of a regular summation. Consequently, their boolean product $W \odot H$ results into binary/boolean matrix $A_{bool} \in \{0,1\}^{m\times n}$. Thus their inner-product (row-column) view can be calculated as follows: for each entry a_{mn}^{bool} of matrix $A_{bool} = W \odot H$:*

$$a_{mn}^{bool} = \bigvee_k w_{mk} \wedge h_{kn} = \bigvee_k w_{mk} \cdot h_{kn}$$

The introduction of the OR operation makes it impossible to use this as a metric for real-valued matrices. Therefore, any approach using this metric cannot compare itself with relaxed versions of the problem or use a deterministic lower bound offered by the principal component analysis.

Now we present a lemma that proves the boolean multiplication $W \odot H$ produces matrices with lesser norm than matrices produced by regular multiplication $W \cdot H$. Then we show that this implies that always the Frobenius norm using the boolean multiplication will also be smaller using the boolean matrix multiplication, making it a more forgiving metric.

Lemma 3.8 (The BoolMM lesser norm)

$$\|W \odot H\|_2 \leq \|W \cdot H\|_2 \rightarrow \|A_{bool}\|_2 \leq \|A_{regular}\|_2$$

Proof. Let $A_{\text{regular}} \in {}^{m\times n}$ be the regular matrix multiplication product of $W \cdot H$. Then each entry a_{mn}^{regular} of A_{regular} can be defined as:

$$a_{mn}^{\text{regular}} = \sum_k w_{mk} \cdot h_{kn}$$

Consequently, we divide it into two cases. First, when $\bigvee_k w_{mk} \cdot h_{kn} = 0$, it means that all k pairs of $w_{mk} \cdot h_{kn}$ are also zero, then their summation is zero. Thus in this case, $a_{mn}^{\text{bool}} = a_{mn}^{\text{regular}} = 0$.

In the other case, when $\bigvee_k w_{mk} \cdot h_{kn} = 1$, then at least one of the k pairs of $w_{mk} \cdot h_{kn}$ is one. Thus, in this case $1 = a_{mn}^{\text{bool}} \leq a_{mn}^{\text{regular}} \leq k$.

Finally, if for all $a_{mn}^{\text{bool}} \leq a_{mn}^{\text{regular}}$, and because the norm is a function of the summation over all a_{mn}, then $\|A_{\text{bool}}\|_2 \leq \|A_{\text{regular}}\|_2$. ■

For this reason, we can extend this lemma to the approximation error.

Theorem 3.9 (The BoolMM creates a forgiving metric) *The Frobenius norm error calculated using the boolean matrix factorization is always* *less or equal to the one calculated using regular matrix factorization.*

$$\|A - W \odot H\|_2 \,/\, \|A\|_2 \leq \|A - W \cdot H\|_2 \,/\, \|A\|_2$$

Because the reconstruction $W \cdot H$ can produce entries that are larger than 1*, up to rank k, it penalizes in the norm the effect of over-covering a position in the original matrix A with more than one component.*

Proof. This proof is similar to the argument in the previous lemma. Let each entry of the given matrix A be a_{mn}^{given}. Then the error matrix has entries $a_m^{\text{given}}n - a_{mn}^{\text{bool}}$ in the $A - W \odot H$ case, and $a_m^{\text{given}}n - a_{mn}^{\text{regular}}$ in the $A - W \cdot H$ case.

In the case where $a_{mn}^{\text{bool}} = a_{mn}^{\text{regular}} = 0$ then the term $(a_{mn}^{\text{given}} - a_{mn}^{\text{bool}})^2 = (a_{mn}^{\text{given}} - a_{mn}^{\text{regular}})^2$.

In the other case, when $1 = a_{mn}^{\text{bool}} \leq a_{mn}^{\text{regular}} \leq k$, we have two subcases.

The wrong covering subcase when $a_{mn}^{\text{given}} = 0$ then the error for position m, n is of the BoolMM $(a_{mn}^{\text{given}} - a_{mn}^{\text{bool}})^2 = (0-1)^2 = 1$ whereas for the regular matrix multiiplication $1 \leq (a_{mn}^{\text{given}} - a_{mn}^{\text{bool}})^2 \leq k^2$.

The other subcase is the covering/overcovering, where $a_{mn}^{\text{given}} = 0$. The BoolMM error is also fixed $(a_{mn}^{\text{given}} - a_{mn}^{\text{bool}})^2 = (0-0)^2 = 0$, while the regular matrix multiplication will have error zero if it covers only one time, but will penalize the overcovering of the position $0 \leq (a_{mn}^{\text{given}} - a_{mn}^{\text{bool}})^2 \leq (k-1)^2$.

Therefore, the error is always greater or equal for each entry in the error obtained by the regular matrix multiplication. Then, we can affirm:

$$\|A - W \odot H\|_2 \leq \|A - W \cdot H\|_2$$

And finally,

$$\|A - W \odot H\|_2 \,/\, \|A\|_2 \leq \|A - W \cdot H\|_2 \,/\, \|A\|_2$$

■

Thus, calculating the error using the regular matrix factorization is a more strict metric in the sense that it always indicates a greater error.

Whenever there is an over-covering effect of two or more components covering the same position, the Boolean Matrix Multiplication masks it from the calculation, by setting the reconstruction to one. Even worse, the Boolean Matrix Multiplication doesn't allow for comparing the error loss with continuous methods and bounds.

4
Small Synthetic Cases and Algorithm Comparison

In this chapter, we present three small cases which could be solved easily by hand and paper. Although some solutions can be obvious, the algorithms we compared did not produce correct solutions. Thus, we believe this chapter serves as a side note on the quality of the implementations regardless of the metric used to evaluate it. Also, it raises interesting questions that we will address in the discussion or in future work.

We propose three synthetic cases: A, B, and C. They are 5×5 matrices of rank two, and their factorization is of increasing difficulty for testing some different aspects. Case A tests the ability to detect balanced components. Case B tests the ability to detect unbalanced and not contiguous components. Finally, case C tests the ability to detect components with one position with ambiguous belonging. The importance of this chapter is also to compare some qualitative aspects regardless of differences in error metrics.

For each of these cases, we apply an implementation of the (working) algorithms in R Package in (DESOUKI, 2021), GreConD (BELOHLAVEK; VYCHODIL, 2010), and TopFiberM (DESOUKI; RöDER; NGOMO, 2019), as well as a simplified version of our proposed BackDisc pipeline. In this simplified pipeline, we use thresholds $t_w = .5$ and $t_h = .5$. Also, since Zhang's algorithm (ZHANG et al., 2007) implementation by (ZITNIK; ZUPAN, 2012) is not deterministic and the first step in our proposed BackDisc pipeline, the whole pipeline tends to generate different answers in each run. Thus, we present three runs of our proposed BackDisc pipeline as examples.

4.1
Case A

In this case, we test if these factoring algorithms can catch two balanced components using rank-2. The components should weigh the same in the Frobenius Norm, and they are contiguous. Also, there is no overlap in entries; every entry belongs to a single square pattern.

Therefore, this case should ideally produce the following factorization. The other alternative would be a permutation of the components, that is, columns of the mixture matrix W and the factor or dictionary matrix H.

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \xrightarrow[\text{ideally}]{} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

4.1.1
Case A - GreConD

$$\xrightarrow[\text{GreCond}]{\text{BMF}} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4.1.2
Case A - topFiberM

$$\xrightarrow[\text{topFiberM}]{\text{BMF}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4.1.3
Case A - Our BackDisc Pipeline

$$\xrightarrow[\text{Zhang+BackDisc run 1}]{\text{BMF}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\xrightarrow[\text{Zhang+BackDisc run 2}]{\text{BMF}} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\xrightarrow[\text{Zhang+BackDisc run 3}]{\text{BMF}} \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

In this first case, we observe that both heuristic algorithms were able to detect only one of the components, while our proposed BackDisc pipeline could detect both of them in two out of three runs. GreConD also duplicated the column in the first matrix, while the associated row in the second matrix had entries equal to zero.

4.2 Case B

This case is more complicated because one of the components does not add much to the error being minimized. Thus, an algorithm can ignore it as an outlier and not take advantage of the space for a second component to capture the pattern perfectly.

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \xrightarrow[\text{ideally}]{} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

4.2.1 Case B - GreConD

$$\xrightarrow[\text{GreCond}]{\text{BMF}} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

4.2.2 Case B - topFiberM

$$\xrightarrow[\text{topFiberM}]{\text{BMF}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

4.2.3 Case B - Our BackDisc Pipeline

$$\xrightarrow[\text{Zhang+BackDisc run 1}]{\text{BMF}} \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\xrightarrow[\text{Zhang+BackDisc run 2}]{\text{BMF}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\xrightarrow[\text{Zhang+BackDisc run 3}]{\text{BMF}} \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

This case has proven to be harder. As expected, all algorithms produced wrong answers, with the exception of our proposed BackDisc pipeline getting it right in the last run. At least, the algorithms prioritized covering the larger pattern most of the time, but they were not able to consistently take advantage of an extra component to capture the smaller pattern.

4.3 Case C

This case tests the ability of the algorithms to capture a pattern with a shared position. This should show the intricacies of their behavior because GreConD in (BELOHLAVEK; VYCHODIL, 2010) and TopFiberM (DESOUKI; RöDER; NGOMO, 2019) use the reconstruction of their factorization using the Boolean Matrix Multiplication, so they should disregard the problem of overcovering. In contrast, our proposed BackDisc pipeline minimizes the reconstruction through regular matrix multiplication but a linearized version of the Frobenius reconstruction error minimization. This version penalizes the effect of overcovering, and we wanted to verify if this aspect would make the response more sparse.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \xrightarrow[\text{ideally ??}]{} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

4.3.1 Case C - GreConD

$$\xrightarrow[\text{GreCond}]{\text{BMF}} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 2 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

4.3.2 case C - topFiberM

$$\xrightarrow[\text{topFiberM}]{\text{BMF}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4.3.3 Case C - Our BackDisc Pipeline

$$\xrightarrow[\text{Zhang+BackDisc run 1}]{\text{BMF}} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\xrightarrow[\text{Zhang+BackDisc run 2}]{\text{BMF}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\xrightarrow[\text{Zhang+BackDisc run 3}]{\text{BMF}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

TopFiberM also discovered one of the two patterns in this case, just as it has in the last case. GreConD had interesting behavior in this case; it not just was not able to detect the second component in full, but it also overcovered using an unnecessary dense second component.

Our approach got it right the third time with the expected overcovering in the central position. The other responses were surprisingly close. The first response was also dense but did not overcover three positions like GreConD did. The second response was a choice of not covering two positions instead of overcovering the central one.

All this variability ties back to the choice of error metric to be minimized. GreConD did present a significant unnecessary intersection; this must be linked with not penalizing overcovering by using the boolean matrix multiplication.

5
Main Application: Process Mining for Petrochemical Batch Processes

In petrochemical batch processes, the state of the valves is representative of the state of the whole process. Certain combinations of opened and closed valves are necessary to enable process phases and their objectives. Although the operational procedures are well known, there are situations where the operation must be slightly different. It is vital to detect and document those variations. Thus, there is a great need for automatic visualization of the process for training and supervision.

There are process discovery algorithms that can use the log of events in the process and try to describe them as flowcharts (AALST; AALST, 2011). The problem is that in modern petrochemical batch processes, hundreds of valves generate opening and closing events, and there is no indication of when the process returns to the same state as before. Therefore, the standard algorithms are inadequate to deal with this setting.

Process discovery algorithms need a clear definition of a case's beginning and ending. Petrochemical batch processes are recurrent, approximately returning to the same state in a predetermined cycle duration. Therefore, it is another challenge to identify precisely when the process returns to the same state as before, without any prior knowledge of the process or the installation specificities of each refinery. The slight variations in batches can also haste or delay the end of processing a specific batch.

Moreover, process discovery algorithms have difficulty working with hundreds of events because they rarely have enough order to create cohesive flowcharts (AALST; AALST, 2011). It is more likely to create "macaroni" flowcharts, with hundreds of boxes connecting themselves with thousands of arcs. Therefore, standard algorithms are inadequate to deal with this setting and produce understandable process representations.

Our approach deals with these problems separately, using two matrix decomposition techniques. First, we use the principal component analysis (PCA). If there is a strong cyclical behavior, it shows in one of the first columns of the factorized matrices. It is possible to use this factorization to detect which timestamps the process returns to the same state. It is also possible to compare in the reduced space if processes were similar enough to configure a cohesive modus operandi.

Then, we use binary matrix factorization (BMF) to accomplish two tasks:

discover which groups of valves tend to be opened at the same timestamps and which groups discovered this way were opened at each of the timestamps. Instead of working with each event of opening and closing of individual valves, we propose to work with discovered alignments made of groups of valves. In this chapter, we compare the Frobenius Norm Error and the time it took to produce the results of the BMF algorithm using the standard implementation of the BMF algorithm in the Nimfa package (ZITNIK; ZUPAN, 2012) and discretizing its results by two methods.

We compare discretizing it with our custom formulation for using the Gurobi solver, presented in 3-3 and compare it to using our proposed BackDisc pipeline and the BackColumn presented in chapter 3. We also compare these results of the BMF algorithm using the GreConD and TopFiberM algorithms.

Both factorizations, BMF and PCA, are the fundamental step to a new pipeline for solving this and produce understandable representations of the process.s problem. We first describe the steps of this pipeline, present the measurable algorithm results for the sake of comparison in the context of this thesis, and then present the Gantt charts produced to visualize the process in different periods of time in two different refineries.

5.1 Pipeline

Our objective is to visualize batch processes where the main indicators of state are which valves forming alignments are open. Our proposed pipeline bifurcates into two branches then they merge for the final plotting of the batch process representative Gantt charts.

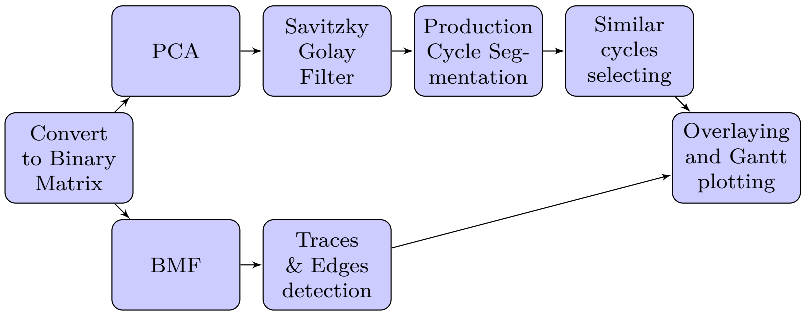


Figure 5.1: Our proposed Gantt pipeline for obtaining Gantt charts using PCA and BMF

First, we apply the principal component analysis and smooth the compressed signal through time using a polynomial filter using the Savitzky-Golay filter (SAVITZKY; GOLAY, 1964). The smoothed signal is periodic, and its local maxima, local minima, and points of intersection in the $y = 0$ line often indicate turning back to the same state. We chose to define a production cycle between the times where the smoothed projected curve intercepts the $y = 0$ line. Then the curve segments can be compared to each other to find similar cycles.

The PCA also allows us to compare time periods within a larger dataset in which the process behaved more consistently. We must select a series of contiguous production cycles with a similarity measure above a certain threshold. Then, we can overlay the production cycles and plot the Gantt chart of the process.

Also, in the other branch, we cluster valves into groups and determine when these groups were used in time using the binary matrix factorization (BMF), simple thresholding, custom mixed-integer formulation, or using a commercial mixed integer solver or our proposed BackDisc pipeline using the BackColumn procedure for its discretization, and reconstruction error recovery. Then, we detect sequences of groups with temporal relationships using a custom mixed-integer formulation or an adaptation of the BackColumn procedure. Lastly, we post-process all these pieces to obtain a full representation of the Plant as a Gantt Chart.

In this setting, we define $\mathcal{V}$ as the set of valves to be monitored, and we define $\mathcal{T}$ as the set of timestamps in which we sample the state of valves. We also define matrix $A_{[\mathcal{T}\times\mathcal{V}]}$ as the matrix with binary entries $a_{tv} \in \{0,1\}$. An entry a_{tv} equals one if the valve v was open at timestamp t.

In figure 5.3, we present intermediate results of both factorizations in parallel. The PCA decomposes the valves samples matrix A into $U_k \cdot \Sigma_k \cdot V_k^T$, and we plot the first column of U as the signal that is most representative of the valves states in time. As we can observe, it is periodical. We smooth it and segment it in time. In the same figure, we present below the binary columns of matrix W obtained using BMF. They represent groups of valves used through time.

Some of the aspects appear very clear in this figure, such as the production cycles representation, their similarity, and also when this similarity streak is broken at the middle of the horizontal axis (time) representation. We can observe a longer cycle in the middle of the example that breaks the timeline into two periods with internal cohesion.

We discuss each of these Gantt pipeline branches in detail in the following sections.

5.1.1 Principal component analysis (PCA) for cycle segmentation and temporal cohesiveness selection

The PCA is an algorithm for capturing greedily the variance of a given matrix, constructing the best low-rank approximation rank by rank. Let $A_{[\mathcal{T}\times\mathcal{V}]}$ a real-valued matrix, and G a desired rank of approximation, the PCA algorithm returns orthonormal matrices $U_{[\mathcal{T}\times\mathcal{G}]}$ and $V_{[\mathcal{V}\times\mathcal{G}]}$, and a diagonal matrix Σ that when multiplied as in $U \cdot \Sigma \cdot V^T = A_{\text{approx}}$. This decomposition can be used for signal/image compression, embedding observations in a lower dimensional and orthogonalized space, and enhancing the predictive power of simple machine learning methods such as decision trees (WOLD; ESBENSEN; GELADI, 1987). In matrix V, the most frequent patterns of opening of valves are captured, and in matrix U, the oscillations in the state of the industrial Plant are compressed.

The outer product of the first column of U and the first row of V^T scaled by the element σ_{11} of Σ is the best approximation of rank one to the original decomposed matrix. Since matrix U and V can have positive or negative entry values, it is common in some cases for the first principal components to assume two common patterns, one using positive values and another using negative values. Consequently, it captures two opposing patterns of opening of valves in the same component, expressed in the first row of V^T.

For instance, if we process the training data of the famous hand-written digit recognition dataset MNIST from Lecunn in (LECUN, 1998), we observe that the first principal component represents the hand-written zero pattern using positive values and the hand-written ones pattern in the negative direction. It captures two opposing patterns in the same component, expressed in the first row of V^T. Regions are more present in the written digit 0 as positive in orange and regions of the written digit 1 as negative in purple. We demonstrate this effect in figure 5.2.

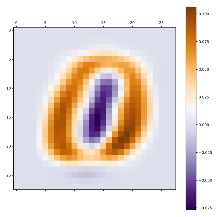


Figure 5.2: Principal component of the training data in the MNIST dataset as an intuition.

Therefore, in cyclical settings, we observe that the first principal component in the first row of V^T tends to represent a pattern of the valves of the

Plant on the positive side and another pattern on the negative side. Consequently, in the first column of matrix U, we observe the effect of a temporal oscillation between these patterns, revealing both the duration of and similarity between production cycles. At the top part of figure 5.3, we see plotted in blue a segment in time of the first column of matrix U. The pattern is almost periodic, although it has a strange shape. We applied a polynomial filter, the Savitzky-Golay smoothening technique (SAVITZKY; GOLAY, 1964), to obtain a curve with less noise. The smoothed curve is represented as the dark yellow curve at the top of figure 5.3.

We established a time segmentation rule the instant the smoothed curve crossed the origin ($y = 0$) going down, with a negative derivative. Figure 5.3 represents this segmentation with red vertical lines. Temporal segmentation is used to later segment the temporal components of both factorizations into production cycle chunks. Then, consecutive chunks can be compared to one another. The PCA's first column of the compressed space $U_{[\mathcal{T}\times 1]}$ can be segmented into segments. Suppose the segmentation rules identified a set $\mathcal{S}$ of production cycle transitions at timestamps $t_1, t_2, \ldots, t_{\mathcal{S}}$. Between these timestamps, we define segments of vector $U_{[\mathcal{T}\times 1]}$ as $C_s = U_{[t_s:t_{s+1},1]}$. These segments capture the variation between opposing plant states described by the positive and the negative part of the first row of matrix V^T during the same cycle.

These segments can be compared to check the similarity between cycles. We define the distance between cycles as the normalized Frobenius norm of the difference between cycles $\|C_s - C_{s+1}\|_2 \,/\, \|C_s\|_2$. In the segments that do not have the same length, we pad the shortest one with entries equal to zero. The main idea is to filter contiguous intervals of time in which this difference between cycles is within a reasonable threshold. In figure 5.3, we can notice visually that there is a change in pattern from the seventh segment forward. For the final purpose of visualization of the production cycle, big changes in the process introduce noise that perturbs the chart. If the process suddenly goes through a change, the method will have a more difficult task of depicting two production patterns. Hence, we propose a method for selecting cohesive cycles.

Also, the processing time of subsequent steps of the algorithm, in special the Binary Matrix Factorization, is very high for more than a thousand timestamps. Therefore, we only run the next steps in selected segments of a large dataset. The Frobenius norm between cycles allows us to identify periods with internal consistency and run the next steps separately.

5.1.2 Binary Matrix Factorization (BMF) for valve clustering and detecting the use of groups of valves in time

Given a matrix with binary entries $A_{[t\times v]}$, the binary matrix factorization obtains ideally two also binary matrices $W_{[t\times g]}$ and $H_{[g\times v]}$. Each row of matrix H will represent groups of valves that tend to open together, potentially representing alignments. Each row of matrix W will represent the timestamps to which the corresponding groups of valves were open.

We obtain a first approximation of the factorization using the gradient descent algorithm proposed by Zhang et al. in (ZHANG et al., 2007) and implemented by Zitnik and Zupan in the python package Nimfa (ZITNIK; ZUPAN, 2012). The algorithm outputs two matrices $W \in [0,1]^{|\mathcal{T}|,|\mathcal{G}|}$ and $H \in [0,1]^{|\mathcal{G}|,|\mathcal{V}|}$, such that the multiplication $W \cdot H$ is an stable local optimum approximation for A.

Therefore, both our proposed BackDisc pipeline or a modern mixed-integer solver with our formulation 3-3 can obtain a truly binary matrix H, using the fixated $\overline{W}$ obtained by Nimfa then thresholded, and later fixating the value of H obtained in the first optimization of formulation 3-3 and solve for W.

Some performance issues may arise when running the formulation. The running time of the factorization is increased greatly for matrices of more than hundreds of columns and rows, more so for a number of groups greater than 20. We compare the discretization time using the formulation and our proposed BackDisc pipeline.

We present a transposed view of the resulting matrix W at the bottom of the figure 5.3. In this example, we indicate an entry $w_{tg} = 1$ of matrix W in yellow if a group of valves g was open at time t and in purple otherwise. Notice that there is temporal cohesion. Groups of valves tend to remain open for several hours, so if w_{tg} is one, then $w_{(t+1)g}$ will also be if group g remained open. Also, notice that some groups of valves tend to have a sequential pattern, that is, one closes, and consequently, another opens. When these patterns are too frequent, groups may have a sequence relationship. We will discuss this in the next section.

5.1.3 Trace detection

At this point in our proposed BackDisc pipeline, we also have the option to detect relationships between the groups of valves in such a way that indicates a sequence of behaviors, a group that usually follows another.

The major challenge to detect this is that processes can have operations in parallel. Whenever two groups of valves are open simultaneously in a row of W, there will be more than one entry equal to one. If there is a sequential relationship between groups, that means when one group closes, the other opens. We have to exclude parallel groups in cases that, by coincidence, open at the same time as the other but carry no real sequential relationship.

We select columns of matrix W in such a way that they approximately cover all timestamps of the set $\mathcal{T}$. So we define the binary vector $x \in \{0,1\}^{\mathcal{G}}$ as the vector that selects columns of W. Ideally, all groups of the factorization would exactly be open when the other closed, then for all instants t, the sum $\sum_g^{\mathcal{G}} w_{tg} x_g = 1$. But any misalignment would deem the selection of groups infeasible. We want to approximately cover all instants t. Thus, we want to minimize the function $|W.x - \mathbb{1}|$, where $\mathbb{1}$ is the vector of all entries equal to one of size $\mathcal{T}$.

We can eliminate the modulus operation by using a mixed integer solver. Therefore, we introduce slack variables r_t^+ and r_t^- that will assume a positive value every time that a time t was not covered or many groups overcovered. In 5-1, we show the MIP formulation.

$$\underset{x_g, r^+, r^-}{\text{minimize}} \sum_{t \in \mathcal{T}} r_t^+ + \sum_{t \in \mathcal{T}} r_t^- \tag{5-1a}$$

subject to:

$$(\sum_{g \in \mathcal{G}} W_{tg}.x_g) + r_t^+ - r_t^- = 1, \qquad \forall t \in \mathcal{T} \tag{5-1b}$$

$$r_t^+, r_t^- \geq 0, \qquad \forall t \in \mathcal{T} \tag{5-1c}$$

$$x_g \in \{0,1\}, \qquad \forall g \in \mathcal{G} \tag{5-1d}$$

The output x will indicate a selection of groups. If we filter matrix W to contain only the columns selected by x, we should be able to perceive a transition between groups of open valves. At most instants t, a row of this filtered matrix W_x should have only one entry equal to one. A simple algorithm can loop through all instants t in time and see if there was a change in the opened groups compared with instant $t+P$ with period P ahead.

This algorithm checks for time t, which is the entry equal to one; then we repeat the procedure for instant $t+P$. Therefore $G_{\text{before}} = \text{argmax}(W_t)$ and $G_{\text{after}} = \text{argmax}(W_{t+P})$. If G_{before} and G_{after} are different, we include the direct relation $G_{\text{before}} \rightarrow G_{\text{after}}$ and its timestamp t into a list.

For a good solution, we consider selections of groups that cover almost all timestamps. If the objective function of the problem 5-1 will be approximately close to the sum of timestamps not covered to the time stamps over-covered. We can normalize this objective function by the number of timestamps $\|\mathcal{T}\|$ in order to make sense of the proportion of defective timestamps. Any optimal x can be evaluated by the percentage of defective timestamps; we established a threshold of 10%. So it is possible for more than one solution to be below this threshold. The presence of more than one solution within the threshold is an indicator of parallelism in the process.

Consequently, we devised a mechanism to generate all the feasible solutions within the threshold. After the first round of optimization, if we found a solution with a percentage lower than the threshold, we added the solution to a list, detected transitions using the corresponding set of columns, and then inserted it into the formulation 5-1 a new constraint that prohibits the solver to output it again. Formally, if the previous round gave solution $\overline{x}$ we inserted to formulation 5-1 the constraint $\sum_{g \in \overline{x}} x_g \leq |\overline{x}| - 1$. Then we stopped at the first solution with the percentage above the threshold.

This formulation can also be adapted to be solved by the BackColumn algorithm. If we notice, minimizing $|W.x - \mathbb{1}|$ is the same as fixating the matrix W and solving a column of all ones $\mathbb{1}$. The columns_solve algorithm just needs to be adapted to receive a threshold of the desired objective function, then the mechanism to save the best solution stores all solutions which have the objective less than the saving threshold.

5.1.4 Gantt chart

In figure 5.3, we show an intermediate representation of the pipeline to a refinery delayed coker unit. This represents the results of the methods proposed in subsections 5.1.1 and 5.1.2. We observe the oscillating pattern of the PCA through time and the corresponding time segmentation in red vertical lines.

The bottom picture corresponds to the transposed view of matrix W. Each of the fifteen rows represents a group of valves that act together, represented in yellow when the group is opened and in purple when it is closed. The main idea of the Gantt chart is to segment matrix W through time using the segmentation detected using the PCA and combining these submatrices.

We use the production cycle segments of matrix W and combine them by simple summation and thresholding. We pad with zeros smaller W segment submatrices until every segment representing a single production cycle contains the same number of timestamps (rows). After that, we sum element-wise all the segment submatrices, obtaining a matrix that corresponds to the aggregate behavior of all the production cycles in the time period chosen. Then we take the median among the entries of the resulting matrix, and for all positions above the median, we set to 1 and 0 otherwise.

5.2 Experiments

To test our proposed Gantt pipeline, we obtained data from two delayed coker units of a large petrochemical company in Brazil. For each refinery, we obtained data on the state of valves related to the batch processes inside the delayed coker unit. These datasets contained samples from the valve states with 30 minutes between samples during approximately one year, which translated to approximately 17 thousand samples or timestamps. Table 5.1 shows some characteristics of the instances.

The delayed coker process is a semi-batch process in which heavy residues of oil are thermally cracked and then converted into solid Coke and vapor inside coke drums. The vapor is then converted into more profitable liquids (Naphtha, Gasoil) and hydrocarbon gases. (SAWARKAR et al., 2007). For the process to be continuous, it is usual to set pairs of alternate Coke drums. When one of the drums is cooling, and the heavy Coke is purged, the other drum is filled up until the middle of the production cycle, when the input stream is switched again.

Now we present in table 5.1 the characteristics of the datasets. The first rows show the beginning and end of the dataset, the number of timestamps, the number of valves, the number of cycles, the duration of each cycle, and

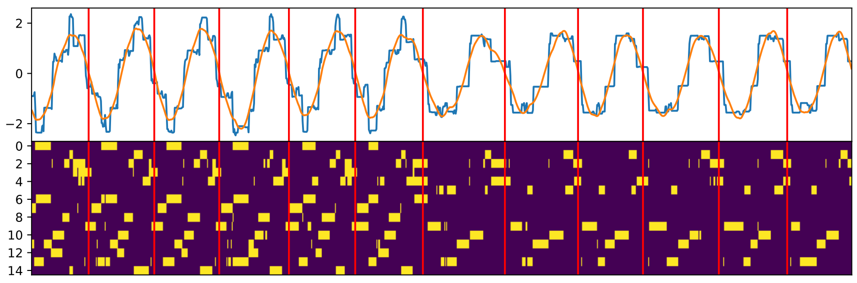


Figure 5.3: The Gantt pipeline's intermediate representation, in which the X axis is time spanning a duration of 25 days. At the top, in blue is the line chart of the first column U_1 in the transformed space by PCA, and in gold is a polynomial smoothing filter of this column U_1. At the bottom is the transposed view of matrix W, where entries $w_{tv} = 1$ are represented in yellow or purple otherwise. The red vertical lines represent the segmentation of production cycles in time.

the number of time periods chosen. The time periods chosen are the number of segments of the dataset that we obtained using the PCA and their median length in hours. The last row shows the number of timestamps in each time period. We chose the time periods with at least five cycles to ensure that the time periods were internally consistent and had a minimum number of 5 similar and contiguous production cycles.

At first glance, the PCA over the matrices for the whole year showed production cycles in the form of recurring patterns. However, it also showed that some recurring patterns changed their behavior from time to time. In a real setting, a user is likely to choose a time period with high cohesion for obtaining a clear picture of the operations during the time period. For that reason, we established a time period selection rule for working separately with time periods that were internally consistent.

We select time periods containing at least five cycles that have the normalized Frobenius norm distance between consecutive periods lower than

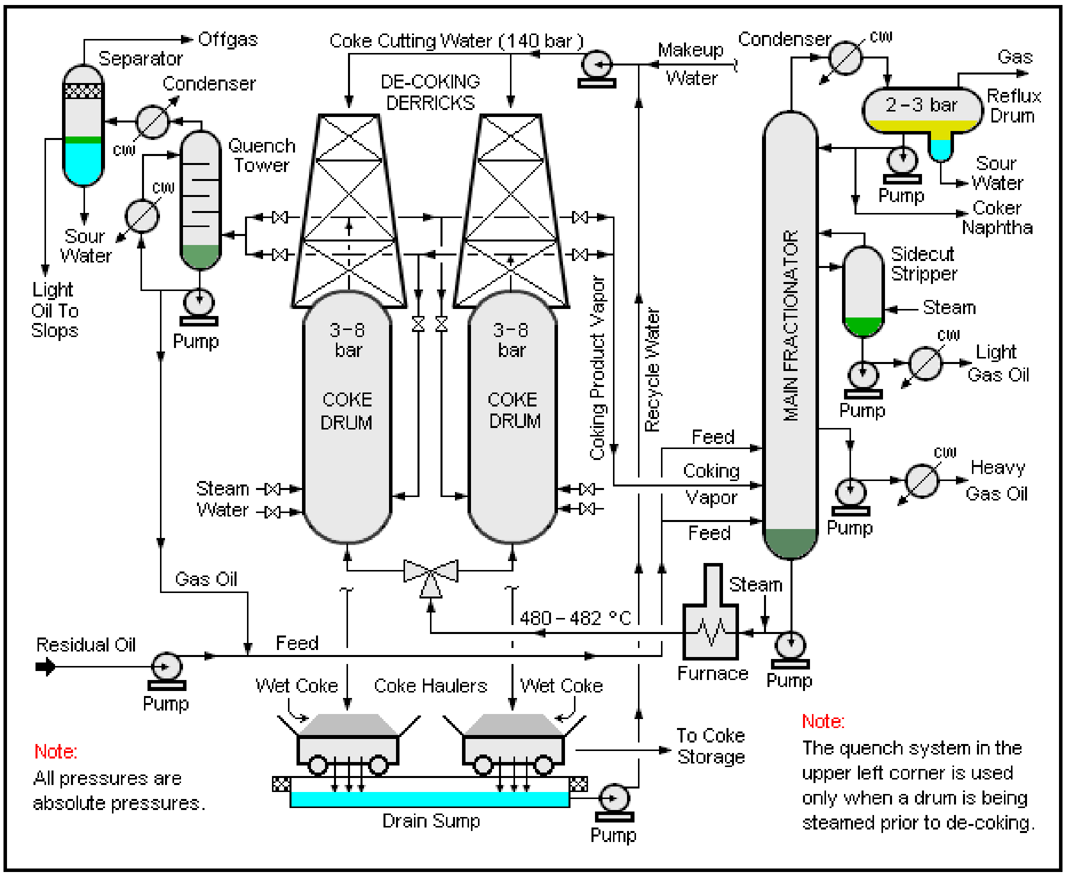


Figure 5.4: Representation of a delayed coker unit by (COMMONS, 2007). It is a semi-batch process in which heavy oil residues are thermally cracked and then converted into solid Coke and vapor inside coke drums in the center. Then, a continuous stream of heavy oil is fed into a switch alternating between the two coke drums.

characteristics	**Refinery 1**	**Refinery 2**
beginning	Jan/2019	oct/2021
end	dec/2019	sep/2022
timestamps	17529	16974
valves	96	52
cycles	168	256
cycle duration (h)	48	32
time periods chosen	6	11

Table 5.1: Datasets characteristics

50%. We considered any distance above this threshold as a sign of discontinuity between cycles.

Using these rules, we selected 6 periods from refinery 1 and 11 periods from refinery 2. For each of these periods, we ran the BMF proposed pipeline. First, we decomposed it using Zhang's (ZHANG et al., 2007) implemented in Python package Nimfa (ZITNIK; ZUPAN, 2012). Then we search for thresholds to approximate a binary matrix, obtaining approximation $\tilde{W}$. And finally, we post-process the resulting matrix using the formulation proposed in 3-3. In both PCA and BMF, we worked with the number of groups $|\mathcal{G}|$ equal to 15.

We measure performances in terms of time in seconds and reconstruction error, using the Frobenius norm distance $\|A - W \cdot H\|_2 \,/\, \|A\|_2$. The reconstruction error of the PCA is used just as a measure of the best reconstruction possible using a rank of 15 if the problem wasn't constrained to produce binary matrices.

We used a PC running Linux, processor 4$\times$ i7-4500U at 1.8 GHz, and 8 Gb of RAM for running this experiment. Specifically, for solving the BMF post-processing formulation and the trace detection formulation, we used Gurobi Optimizer version 9.1.2 build v9.1.2rc0. We show the results in table 5.2.

The thresholding search step in all tests was below a fraction of a second, so we omitted it from the table. What we can observe is that not only the proposed formulation in equations 3-3 obtains truly binary matrices, it is a better alternative than simply thresholding. The thresholding procedure deteriorates the reconstruction error deeming the resulting matrices not adherent to reality.

5.2.1 Results

Our proposed formulation was able not only to recover the error loss that came with thresholding but also could slightly reduce the error compared to Zhang's algorithm, which is counter-intuitive. If the problem was convex or

Instance			**Time** (mm:ss)			**Error** (Frob. Norm)					
Ref.	**T**	**cycles**	**Zhang**	**Gurobi**	**BackDisc**	**PCA**	**GreConD**	**topFiberM**	**Zhang**	**Thres**	**Gur./BackDisc**
1	863	9	00:02	03:22	<0:01	0.08	0.84	0.92	0.47	0.68	0.44
1	684	7	00:01	02:33	<0:01	0.06	1.09	0.85	0.41	0.63	0.36
1	578	6	00:01	02:09	<0:01	0.08	0.92	0.92	0.47	0.65	0.42
1	572	6	00:01	02:08	<0:01	0.07	1.08	0.93	0.47	0.69	0.42
1	1455	15	00:02	05:27	<0:01	0.09	1.02	0.92	0.48	0.70	0.43
1	719	7	00:02	02:41	<0:01	0.10	1.05	0.90	0.48	0.68	0.45
2	528	8	00:01	01:05	<0:01	0.03	1.11	0.93	0.43	0.60	0.37
2	833	13	00:01	01:41	<0:01	0.03	1.29	0.89	0.37	0.57	0.29
2	1475	23	00:02	03:00	<0:01	0.02	1.25	0.89	0.38	0.56	0.26
2	2561	40	00:03	05:09	0:02	0.03	1.21	0.91	0.40	0.60	0.28
2	3036	47	00:04	06:05	0:02	0.05	1.10	0.93	0.43	0.64	0.31
2	1266	20	00:02	02:35	<0:01	0.03	1.16	0.90	0.39	0.59	0.33
2	388	6	00:01	00:47	<0:01	0.02	1.21	0.90	0.37	0.58	0.25
2	1811	28	00:03	03:40	0:01	0.04	1.14	0.90	0.43	0.63	0.35
2	983	15	00:02	01:59	<0:01	0.02	1.18	0.91	0.39	0.59	0.26
2	679	11	00:01	01:23	<0:01	0.02	1.19	0.92	0.40	0.58	0.29
2	1357	21	00:02	02:44	<0:01	0.03	1.08	0.94	0.42	0.65	0.35

Table 5.2: Runtime performance metrics. We omitted the benchmark running times. All of the GreConD were below 1 per second, and all topFiberM were all below .1 per second.

near convex, we would expect the gradient descent approach to always have a better construction error than the strictly binary one because the decision space of the relaxed problem, allowing for any value between 0 and 1, is larger and contains the decision space of strictly binary matrices. That indicates that this implementation of the gradient descent algorithm halts at a very low-quality local optimum.

Finally, we subdivide matrix W into production cycles and combine the production cycles in order to produce the Gantt chart. Also, we adjust the timestamps of the detected hand-offs for the period, calculating the time delta of when it happened since the beginning of the production cycle. Usually, almost all production cycles have the same hand-offs at approximately the same point in time.

So we take the median of these time deltas to position the upwards or downwards arrows on the x-axis. We present the final Gantt chart for the two refineries in selected periods in figures 5.5 and 5.6. We present all the Gantt charts for all the selected periods in the appendix A.

The PCA approximates the singular value decomposition. Thus, it has the best low-rank representation of the given matrix. Consequently, we observe that for the chosen rank of 15, the reconstruction error of the PCA lower bounds is possible to obtain with a more constrained factorization such as the BMF.

Thus, the PCA served as a lightweight building platform to detect the oscillating patterns of the production cycles and the detection of discontinuities in how the process was operated. We devised an algorithm that can suggest periods in time with low variability to help the subsequent steps of our pipeline work with less noise.

We could detect that the process usually had cycles of approximately 48 hours and a higher variability between cycles in refinery 1. And in refinery 2, cycles with approximately 32 hours and lower variability between cycles. These production cycle lengths were later confirmed with personnel directly involved in each refinery.

Our proposed BackDisc pipeline presented in 3 and our formulation in equations 3-3, could post-process the approximation obtained by gradient

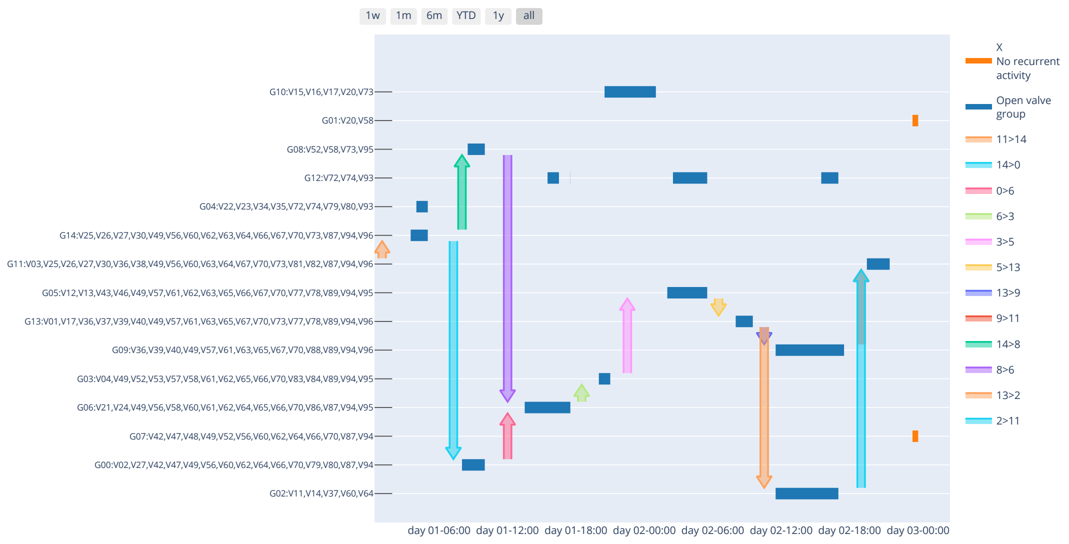


Figure 5.5: Our pipeline's final representation, the Gantt Chart. Lines indicate groups of valves, and arrows indicate frequent maneuvers to change between groups. Notice that the chart is a generic representation of the state of groups of valves combining six 48 hours cycles in refinery 1 during the fourth selected time period.

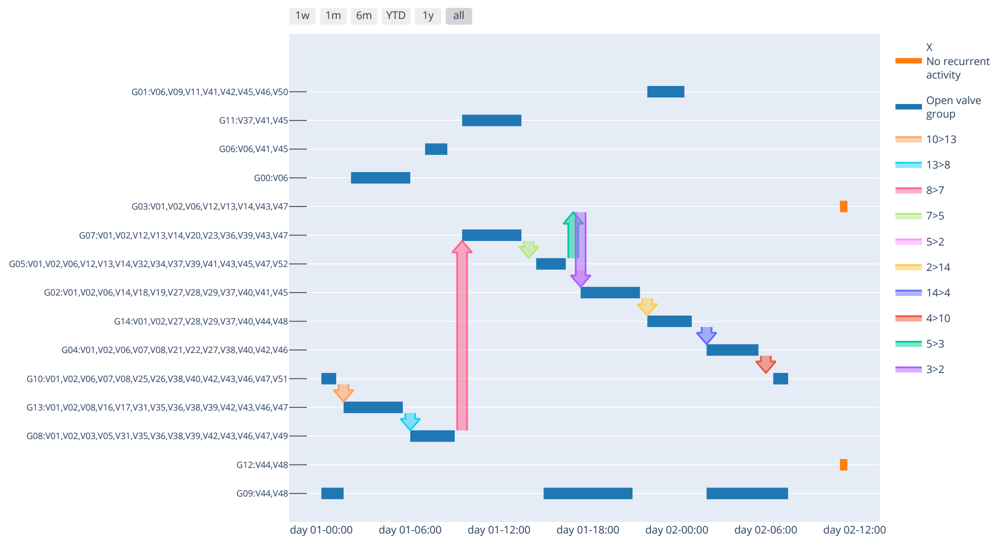


Figure 5.6: The same chart for the fifth selected time period in refinery 2. The chart is a generic representation of forty 32-hour production cycles.

descent methods discretizing them to strictly binary matrices, recovering the great error loss that the greedy thresholding approach introduces.

An unexpected effect was to reduce the error loss to a point even lower than the gradient descent methods. This represents that the bilinear aspect of the relaxed problem must generate local optima that halt gradient descent methods such as (ZHANG et al., 2007) at suboptimal solutions. The other algorithms did not produce representations even close to being representative of the original matrix, achieving error levels above 90%; our algorithm could achieve error levels below 40%.

Before we proposed this novel discretization approach for the BMF, the Gantt pipeline would choose between working with non-binary values and thresholding. The user would have to interpret real values for both entries h_{gv}, which would be the strength of the belonging to a group, and real values for entries w_{tg} for the strength of the use of a group of valves. Intermediate values such as .5 would mean that to explain an entry in the given matrix $a_{tv} = 1$, there could be a combination of different groups where the same valve would contribute.

The alternative would be to discretize the resulting matrices, incurring an error loss to a level that deems the reconstruction matrix not representative of the original one. Therefore, exists a complicated trade-off between the interpretability and fitness of the original data.

Finally, observing the resulting Gantt chart, our proposed BackDisc pipeline could detect two behaviors that are parallel in execution during the whole operation. If we observe closely, the hand-offs represented by the vertical arrows form two different traces. Finding two or more traces clearly indicates parallelism in process, which is one of the greatest challenges for process discovery algorithms (AALST; AALST, 2011).

The Gantt chart could also capture the parallel nature of the cooling process in Refinery 1. The installation there consists of two pairs of coke drums that are phased 6 hours from each other. The parallel traces represent the changes in the openings of valves associated with each pair of coke drums. We confirmed with the refinery personnel that the method approximately identified which valve belonged to what drum.

In practice, we presented a theoretical framework that enables the discovery of a petrochemical batch process without any prior knowledge needed. The Gantt chart captures the production cycle length, important valve alignments, their sequence in a trace of events, their length, and their transitions at a glance.

Future research could explore many possibilities. The time represented in

PCA's matrix U could be a strong indication of delays or significant changes in expected steps. Another possibility would be to compare the decomposed matrices of two different time periods to evaluate changes in the process length and the existence of alternative alignment routes.

6
Other applications: Gene Expression, Topic Modelling, Recommendation Systems

In this chapter, we united other direct applications of the BackDisc pipeline that we proposed in chapter 3. We will show how the BackDisc pipeline can be used to solve binary matrix factorization that arises in different domains. First, we present a comparison of our method to other algorithms using their instances but using the Frobenius Norm Reconstruction error using regular matrix multiplication as opposed to boolean matrix multiplication.

Then we run tests in other domains, such as gene expression, topic modeling, and recommendation systems. The intriguing thing about these domains is that they have different sizes, clusterability, and sparsity. We will show that our method is able to handle these different domains and that it is able to recover some reconstruction errors caused by thresholding, even in challenging applications.

Each experiment followed linearly the following steps: (0) using a preprocessing specific to the domain for obtaining binary matrices, (1) decomposing the matrix using Zhang's (ZHANG et al., 2007) algorithm implemented by the python package Nimfa (ZITNIK; ZUPAN, 2012), (2) searching for the combinations of thresholding of the matrices W and H which minimized the reconstruction error, (3) ran the algorithm described in 3.

We designed the experiments to check if this approach would succeed in different settings, both in terms of processing time, reconstruction sparsity and reconstruction error. We measured the processing time of the steps 1, 2, and 3.

Specifically, we measured the times for running our algorithm for optimizing over the matrix W and the matrix H separately. Our implementation had no parallelization since, in the applications, the matrices tend to have an order of magnitude more rows than columns, and the for-loop through rows to obtain the binary version of matrix W usually takes much longer.

Secondly, we measure the sparseness of the reconstruction. One of the aspects we noticed in preliminary tests is that the sparseness of the reconstruction is an influential source of error in the thresholding approach. That means we want to know if the thresholding approach tends to fixate entries to zero in an exaggerated manner. Thus we measure the reconstruction error of the thresholding approach and our algorithm and compare it to the original. Notice that since the gradient-descent approach yields continuous results, it produces a non-binary reconstruction. Then we could not measure its sparseness.

Finally, we measure the reconstruction error using the Frobenius norm as a fraction of the Frobenius norm of the original matrix. Formally, we calculate using the formula $\|A - W \cdot H\|_2 \,/\, \|A\|_2$. This measures how much of the variance of the original matrix did the decomposition capture, even though bounded by rank G.

We compare each step of the algorithm with a lower bound calculated using the matrix's principal component analysis (PCA). The PCA is an algorithm for obtaining a partial or truncated singular value decomposition (SVD), given the desired rank G. The SVD has an interesting property that it has the best reconstruction possible using a given rank G.

Thus it serves as a deterministic and unique lower bound on the problem, sometimes denoting if the problem is even expressible in lower rank factorizations. There are situations, in special in the Recommendation and Topic Modelling, where the algorithm's error is high, but the PCA lower bound shows that the problem was hard to project to a lower rank G.

6.1 Comparison to other datasets in the literature

This section compares GreConD(BELOHLAVEK; TRNECKA, 2013) and TopFiberM (DESOUKI; RöDER; NGOMO, 2019) to our proposed BackDisc pipeline using the same datasets used in (DESOUKI; RöDER; NGOMO, 2019), which are common to many papers in this community. The datasets are Chess and Mushroom found in (ASUNCION; NEWMAN, 2007), DBLP found in (MIETTINEN, 2009), Firewall1 (ENE et al., 2008) and Paleo by (LUCCHESE; ORLANDO; PEREGO, 2010). We test the datasets for ranks 1, 2, 5, 10, and 20 using the Frobenius relative loss using regular matrix multiplication and time in seconds.

Below we present the table with the results.

We can observe that given more room to improve, the metric of the Frobenius loss stays the same for topFiberM and greatly augments in GreCOnD. Our BackDisc pipeline was able to scale to larger ranks taking advantage of the increased space to improve the loss. Possibly because these algorithms are designed to ignore the overcovering loss by calculating it using a specific boolean matrix multiplication. Also, by their greedy construction, they must quickly reach a level of error that they can not later improve upon.

Dataset					time			error			
name	rows	columns	Sparsity	rank	grecond	topfiber	BackDisc	PCA	grecond	topfiber	BackDisc
Chess	3196	76	0.486	1	0.262	0.022	1.884	0.48	0.90	0.62	0.62
Chess	3196	76	0.486	2	0.424	0.022	2.712	0.45	0.81	0.62	0.66
Chess	3196	76	0.486	5	1.185	0.032	5.761	0.37	0.87	0.62	0.58
Chess	3196	76	0.486	10	1.882	0.066	7.899	0.28	1.19	0.62	0.55
Chess	3196	76	0.486	20	3.109	0.092	18.745	0.16	1.66	0.62	0.48
DBLP	6980	19	0.129	1	0.014	0.007	2.360	0.79	0.93	0.90	0.93
DBLP	6980	19	0.129	2	0.034	0.024	2.693	0.72	0.87	0.90	0.83
DBLP	6980	19	0.129	5	0.078	0.019	4.243	0.61	0.73	0.90	0.70
DBLP	6980	19	0.129	10	0.147	0.058	7.513	0.43	0.61	0.90	0.57
Firewall1	365	709	0.123	1	0.315	0.010	0.468	0.47	0.59	0.99	0.52
Firewall1	365	709	0.123	2	0.420	0.013	1.029	0.34	0.44	0.99	0.47
Firewall1	365	709	0.123	5	1.172	0.026	2.070	0.17	0.81	0.99	0.38
Firewall1	365	709	0.123	10	1.399	0.062	3.442	0.11	1.61	0.99	0.32
Firewall1	365	709	0.123	20	2.152	0.129	5.911	0.07	1.61	0.99	0.17
Mushroom	8124	119	0.193	1	0.158	0.049	5.330	0.64	0.93	0.97	0.86
Mushroom	8124	119	0.193	2	0.308	0.051	9.881	0.59	0.88	0.97	0.81
Mushroom	8124	119	0.193	5	1.187	0.084	17.971	0.49	0.77	0.97	0.70
Mushroom	8124	119	0.193	10	2.232	0.155	24.305	0.41	0.85	0.97	0.66
Mushroom	8124	119	0.193	20	4.692	0.277	34.029	0.31	1.33	0.97	0.58
Paleo	501	139	0.05	1	0.025	0.006	0.276	0.91	0.99	1.00	1.00
Paleo	501	139	0.05	2	0.067	0.016	0.494	0.87	0.98	1.00	1.00
Paleo	501	139	0.05	5	0.220	0.029	0.631	0.80	0.95	1.00	0.97
Paleo	501	139	0.05	10	0.219	0.041	1.741	0.74	0.91	1.00	0.92
Paleo	501	139	0.05	20	0.323	0.037	2.157	0.65	0.83	1.00	0.85

Table 6.1: Comparison of time and error between our BackDisc pipeline and the GreConD and TopFiberM algorithms. We apply them to the instances described in (DESOUKI; RöDER; NGOMO, 2019) and the error using the Frobenius norm with the regular matrix multiplication as discussed in chapter 3.

6.2 Gene Expression

The application of Binary Matrix Factorization for Gene expression data was also introduced by Zhang et al. (ZHANG et al., 2010) for analyzing gene expression data following their main paper in (ZHANG et al., 2007).

These datasets come from an innovation in mapping the genome, introduced by the DNA microarrays and the Serial Analysis of Gene Expression. This allowed scientists to analyze thousands of genes in the same study (SCHENA et al., 1995).

We used the Gene Expression Omnibus Dataset Browser (EDGAR; DOMRACHEV; LASH, 2002) and (BARRETT et al., 2012) as the source for our tests. The datasets gather gene expression over specific diseases or conditions. Diseases range from colitis to maternal use of tobacco to a variety of leukemia and cancer diseases. Our choice criteria were selecting datasets related to diseases with the largest amount of samples.

We used the following datasets found in the URL National Center for Biotechnology Information Gene Expression Omnibus Dataset Browser by (CLOUGH; BARRETT, 2016) : GSE11223 (NOBLE et al., 2008);

GSE1133-GPL1073, GSE1133-GPL1074, GSE1133-GPL96 (SU et al., 2004); GSE12417-GPL570, GSE12417-GPL96, GSE12417-GPL97 (METZELER et al., 2008); GSE13355 (NAIR et al., 2009); GSE13576 (MEYER et al., 2011); GSE1726 (MONKS et al., 2004); GSE1888 (DILLMAN et al., 2005); GSE19392 (SHAPIRA et al., 2009); GSE19429 (PELLAGATTI et al., 2010); GSE21521 (HINZE et al., 2010), GSE22845(TASKESEN et al., 2011); GSE27272(VOTAVOVA et al., 2011); GSE27567-GPL1261, GSE27567-GPL570 (LABRECHE; NEVINS; HUANG, 2011); GSE30310 (MORSE et al., 2012); GSE30999 (SUÁREZ-FARINAS et al., 2012); GSE32474(PFISTER et al., 2009); GSE3578 (IWAKAWA et al., 2007); GSE4115 (SPIRA et al., 2007); GSE4290 (SUN et al., 2006); GSE50948 (PRAT et al., 2014); GSE54514 (PARNELL et al., 2013); GSE6919-GPL8300, GSE6919-GPL92, GSE6919-GPL93 (CHANDRAN et al., 2007); GSE755(TIAN et al., 2003); GSE9820(SCHIRMER et al., 2009).

In these datasets, the expressions are real-valued entries. In Creighton and Hanash (CREIGHTON; HANASH, 2003) and in Liu et al. (LIU; CHENG; TSENG, 2013) have a discretization procedure based on the distribution of the entries for each gene. For values at the higher end of the distribution, they considered that a specific sample was up-regulated in that gene.

In contrast, an entry in the lower end was down-regulated. Their datasets had a log-normal distribution, and they picked a threshold under which they would consider that specific gene as down-regulated and another threshold over which the expression would be considered up-regulated.

Our tests' datasets come from a more diverse source from different studies. We verified that some datasets were already normalized. Some were not log-normally distributed. Then we adapted their approach to an affine-invariant approach, choosing up-regulated expressions from the fourth quartile of their respective distribution and similarly choosing down-regulated expressions in the first quartile.

Consequently, we could apply the same procedure to our tests, independent of whether each of the authors that made their data available had different data transforming procedures such as normalization or scaling. This procedure also had the secondary benefit of fixing the datasets' sparsity to 25%.

Therefore, we pre-processed these 31 instances. The resulting binary matrices ranged from 4776 to 54765 rows representing each gene expression and ranged from 158 to 416 columns because instances ranged in number of samples from 79 to 208. Consequently, when our backtracking discretization algorithm (BackDisc) was looping through rows, it had to solve tens of thousands of discrete basis problems using the first procedure in each instance.

With these pre-process steps, we obtain a binary matrix with rows representing genes and columns representing samples and regulation. By construction, the sparsity of these matrices is very close to 25%. We run our experiment in each of these datasets, varying only the rank of the factorization, using ranks $G \in \{10, 20, 50, 100\}$. Also, we defined a time-out parameter because, in some cases, when the rank was set to 100, the procedure ran indefinitely.

We notice that the procedure can take long to post-process, especially when the G is set to 100. We had an experiment cutoff at 1 hour; for ranks of $G = 100$, only six cases finished under the one-hour limit.

We can also observe that the deterioration caused by the thresholding procedure is accompanied by not tackling the sparsity of reconstruction at the same level as the original. The thresholding tends to leave entries equal to zero frequently. Fortunately, our approach recovers the sparsity and the level of the reconstruction error.

We present in 6.1 the summary of the results of the experiment. For the detailed table of results, we refer the reader to the appendix B.

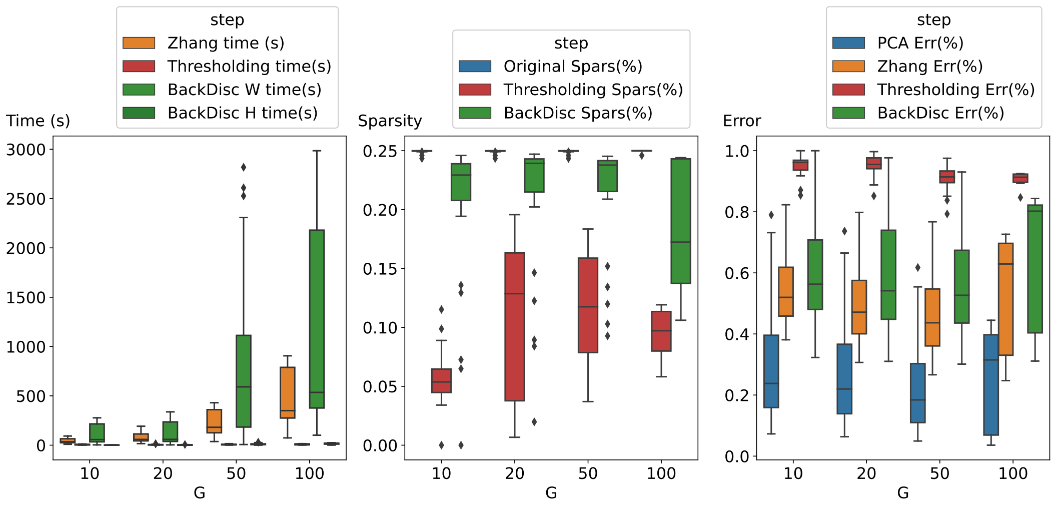


Figure 6.1: Results showing time, sparsity, and reconstruction error of the tests across the 31 instances. In our test, we compare the PCA lower bound in blue, the gradient descent approach described in (ZHANG et al., 2007) and implemented by (ZITNIK; ZUPAN, 2012) in yellow, the thresholding with the minimum reconstruction error found in red, and our backtracking discretization algorithm(BackDisc) in green. The main result is that we can discretize the matrices while maintaining the reconstruction error at the same level as the gradient descent approach; the alternative was to threshold matrices with a significant loss.

However, when comparing the error, we first notice that the lower bound set by the principal component analysis (PCA) is very low, even with a very low rank of $G = 10$, for instance. This shows that the data has some inherent low-rank structure. The error of the gradient descent method is around 50%, that arises to 90% when thresholded and then recovered back to 50% using our approach.

Instances have tens of thousands of rows; for each instance, the time for reconstructing W has to solve tens of thousands of discrete basis problems. Even though, for ranks $G \in \{10, 20\}$, time was under 600 seconds (10 minutes) for all instances, and for ranks of $G = 50$, our algorithm ran under 3000 seconds (50 minutes).

6.3 Topic modelling

A common kind in statistical text mining is encoding documents as bag-of-words (BoW). Bag-of words encoding amounts to determining a shared vocabulary by the documents, excluding rare words and too frequent words (such as prepositions, conjunctions). The basic idea is to select words that encode in their presence a topic in the document. So the basic supposition is that similar words used in documents could cluster under the same topic (GRIFFITHS; STEYVERS, 2004).

Often authors pre-process the words by using just the radical of the words, in a process frequently called stemming or lemmatization (YOGISH; MANJUNATH; HEGADI, 2019). This procedure guarantees that the presence of words with the same radical is counted as the same. Our basic pre-processing was removing stop-words using a list of stop-words in English given by the Python's package NLTK, then removing words that appeared in more than 20% of the documents. Then the basic pre-processing step was to stem all words in documents using the Porter Stemmer (RIJSBERGEN; ROBERTSON; PORTER, 1980).

The datasets used are all available in Python's Natural Language Toolkit (NLTK) package (BIRD; KLEIN; LOPER, 2009). They are:

– **abc** Australian Broadcasting Commission 2006, containing news articles (COMMISSION, 2010)
– **brown** Brown Corpus (FRANCIS; KUCERA, 1979)
– **gutenberg** Project Gutenberg (HART, 2004)
– **inaugural** US presidential inaugural speeches (PRESIDENTS, 1789-2009)
– **movie_reviews** Sentiment data mining (PANG; LEE, 2004)
– **reuters** Reuters financial service articles (WILLIAMS; CALVO, 2003)

– **webtext** Diversed contemporary text genres scraped from the internet (VARIOUS, 2005).

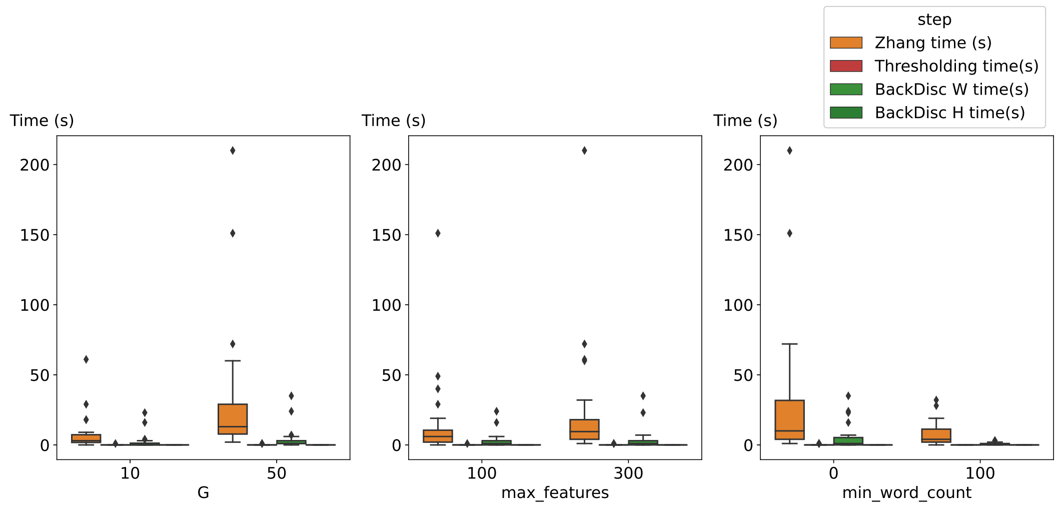
As with the gene expression dataset, we used the rank G of the factorization as a comparison parameter. However, two parameters greatly affect the size and sparseness of the matrices in the Natural Language process setting.

The first one is the size of the vocabulary. We begin adding words from the most to the least frequent until we arrive at a desired number of features we deem sufficient for grouping texts. Increasing too much the size of the vocabulary adds words which are absent from most documents. Later that translates to very sparse columns in the matrix. These very sparse columns do not add grouping information to their problem. We tested two vocabulary sizes: 100 and 300.

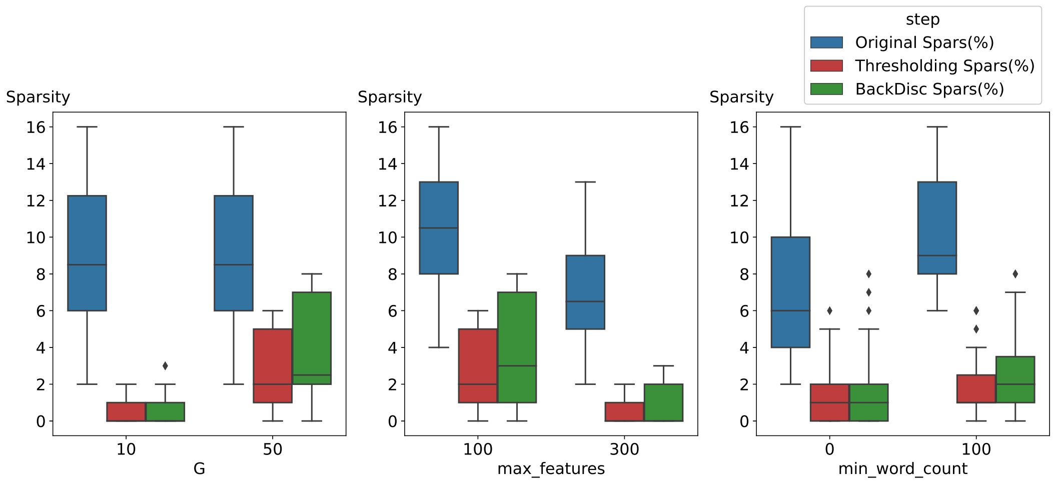
The second parameter is the length of the document. Short documents, when pre-processed, have a great chance of not containing any of the selected words for the vocabulary of the corpus. Thus, we tested the approach also filtering documents of paragraphs too short. Our tests were with no filtering and filtering out documents with less than 100 words.

Therefore, we have three parameters, and we will observe how they affect processing time, reconstruction sparsity, and reconstruction error individually. The full table of results is presented in the appendix B. Here, we present some summary charts of the results.

First, we present the time results, then the sparsity results, and finally, the reconstruction error results. Each of these charts is broken by the main pre-processing premises in order to show the effect of our assumptions on the final result. We compare differences in time, sparsity, and error by the rank of the factorization, size of the vocabulary (max_features), and length of the documents(min_word_count).

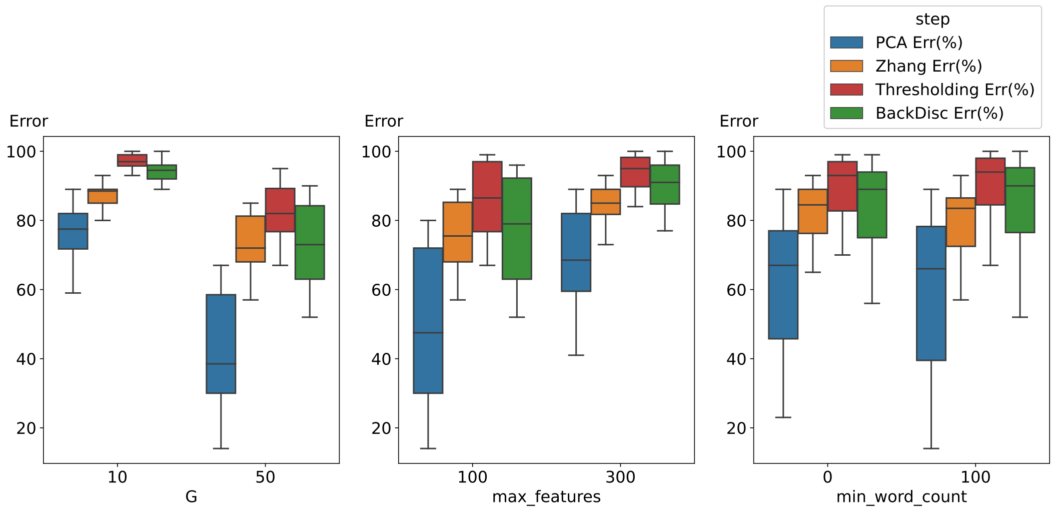


The instances we used in this domain ultimately produced matrices with 100 or 300 columns depending on the vocabulary size and rows ranging from hundreds to a few tens of thousands. Even working with large matrices, all instances took in the W matrix post-processing less than 35 seconds, most in just one second. So, despite the large number of rows, the procedure was able to finish early.



We observe that the original instances are very sparse. Our pre-processing did not allow for stop-words or words that appeared in more than 20% of documents. In the cases where we tested a vocabulary of 100 words, the sparsity varied around 15%. However, in tests with more words, the matrices tend to have an increase in sparseness that negatively affect the error recovery.

Again, just as in the gene expression domain, the thresholding procedure tends to further exaggerate fixating entries to zero. Since the reconstruction possesses significantly more entries equal to zero than the original, we can explain the loss in reconstruction error.



The principal component analysis bound is very high for the test with a low rank of 10, of a level around 80%. When we allow for a higher rank of 50, the PCA lower bound still varies between 20% to 60%. This means that these cases do not have a proper low-rank structure, even if the entries could assume any value. Zhang et al. (ZHANG et al., 2007) gradient descent algorithm achieves decompositions with an error above 70% on most instances; the thresholding approach almost always deteriorates the solution to error levels in around 90%. Our approach recuperated reestablishing previous levels, such as obtained by the gradient descent, with the benefit of delivering truly binary matrices.

An interesting note is that in very few instances, our approach was even better than the gradient descent method, even when navigating a more constrained decision space. Although counter-intuitive, the problem is bilinear, so there must be many local optima in which the gradient-descent approaches get paralyzed during their descent. Our approach, in a few cases, was able to find better solutions.

6.4 Recommendation Systems

The last application domain we tested was recommendation systems. Since the Netflix prize, one very popular approach was to use matrix factorization as a subroutine for matrix completion (KOREN; BELL; VOLINSKY, 2009). Matrices represent in their rows their users and in their columns some features of their behavior in the specific dataset. Some standard features are product visualization, stars for a product review, or simply the purchase/consumption of a specific item.

We chose datasets with various use cases and kinds of features. The datasets are:

– delicious contacts and tags (CANTADOR; BRUSILOVSKY; KUFLIK, 2011), in which we tested both the connection from user to user, which are friends, and the connection from user to bookmarks.

– movie lens 2k (CANTADOR; BRUSILOVSKY; KUFLIK, 2011), which has approximately two thousand users, which gives ratings as they wish to ten thousand movies.

– netflix (BENNETT; LANNING et al., 2007), which also is a dataset of ratings users give to movies.

– Steam purchases (STEAM. . . ,), which we pre-processed to link users to the games they bought respectively.

We took the same intuition we had when analyzing text data. There are many items that are consumed by everybody, and also, there are super consumers who consume a vast amount of the items considered. Concurrently, there are also many items that only a few consumers use and many consumers that do not interact often enough with a base to have a minimal trace of behavior. These aspects translate to dense and sparse rows and columns that do not contribute to define differentiate users amongst themselves of catalog items amongst themselves.

So, we designed a pre-processing that removed columns or rows with more than 90 percent of the entries equal to one, as well as removed columns and rows that were too sparse, with too many entries equal to zero. Even when testing different values for the cutoff percentile, matrices were often too small, or they did not have a pattern perceptible even for a broader decomposition such as the principal component analysis. This approach for controlling the sparsity and size of these matrices is inspired by Diop et al. (DIOP et al., 2019).

Therefore, we selected for factorization datasets and the sparseness controlling pre-processing in which the lower bound on the reconstruction given by the PCA was below 50%. In these test cases, we knew there was minimal room for both gradient descent factorization and our post-processing approach to capture some structure in data. Otherwise, if we tested in cases too sparse or too dense with the PCA error above 50%, we would not even hope for a more constrained setting such as the BMF to capture any structure.

We show the test cases in the table below.

	Original			Test/Filter					PCA	Zhang		Thresholding			BackDisc			
Instance	rows (#)	cols (#)	Sp (%)	Perc. (%)	rows (#)	cols (#)	Sp (%)	G	Err (%)	time (s)	Err (%)	time (s)	Sp (%)	Err (%)	W time (s)	H time (s)	Sp (%)	Err (%)
delicious_contacts	1861	1861	0	50	700	881	1	100	23	18	62	0	0	78	0	0	1	69
delicious_contacts	1861	1861	0	55	520	728	2	100	17	13	58	0	1	75	0	0	1	66
delicious_tags	69223	1867	0	60	17188	746	0	100	48	238	76	0	0	96	3	0	0	97
delicious_tags	69223	1867	0	65	16174	652	0	100	46	180	75	0	0	96	3	0	0	97
delicious_tags	69223	1867	0	70	12076	559	0	100	40	130	72	0	0	95	2	0	0	97
ml_2k	10109	2113	4	75	1722	516	13	100	46	31	76	0	3	94	4	0	4	88
Netflix	480189	17770	1	95	654	863	57	100	43	30	50	0	31	91	29758	89044	44	78
steam_purchases	12393	5155	0	50	5855	2470	0	100	42	246	75	0	0	85	229	0	0	79
steam_purchases	12393	5155	0	70	3344	1490	1	100	39	97	73	0	0	84	321	0	0	77
steam_purchases	12393	5155	0	90	963	513	9	100	27	17	64	0	4	80	179	0	5	71

It is possible to notice the same patterns as before in terms of increased loss using thresholding that can be partially recovered using our post-processing. What stood out in analyzing these instances is that in cases where sparsity was still low after pre-processing, our algorithms could not recover the error. However, in cases where the sparsity was at least above 1%, we could recover some.

In special the Netflix case, it was only after removing 95% of the sparsest columns and rows amount of original rows and columns that we found a case in which the PCA bound was at a promising level. However, the matrix was tiny and dense compared to other tests. Ultimately, our algorithm took less than two days to finish its post-processing. We wonder if that significant increase in process time is due to the density. Maybe denser matrices create difficult possible allocations onto sets internally, meaning a vast combinatorial space.

7
Discussion

7.1
Summary of results

There are many formulations for factorizing binary matrices. Many variant problems are also classified as binary matrix factorization or boolean matrix factorization. Related works use continuous values but with entries near zero or one. Other approaches deal with rank-one factorizations exactly, and some approaches are specialized for the symmetric case of Binary matrix factorization.

There are also different requisites on which aspects a factorization should prioritize for different use cases. Users must be aware of the trade-off between accuracy, scalability and explainability. Our approach was to create a method to adapt the result from continuous algorithms that scale well but faced until now a great loss in accuracy when discretizing.

A continuous near-binary answer is only a little use in some practical cases. If the model is not discrete, it is impossible to work with subsets of items or relationships encoded in the matrix's boolean entries. Discretization is a necessary step to make the model useful. We showed that the discretization is not a trivial step and can be done in a way that preserves the model's accuracy within a reasonable time for factorizations up to rank $G = 50$.

We also presented a much-needed comparison of reconstruction approaches when calculating the error of a factorization. We showed that the reconstruction using the boolean matrix multiplication masks the problem of overcovering because it aggregates different rank-one matrices produced by the outer product of columns and rows with the OR function, as opposed to the simple summation.

By using regular matrix multiplication, we have the additional benefit of using the same error metric for both continuous and binary matrices, besides penalizing overcovering and the effect of producing mutually exclusive components, or at least with minimal overlap. We proved that the error calculated by the boolean matrix multiplication is always lower than the error calculated using the regular matrix multiplication. So, when we compared the reconstruction error in the application, we also used the regular matrix multiplication when compared with other algorithms.

We tested our proposed BackDisc pipeline in many different settings.

We began with three small synthetic examples in chapter 4. We showed that our approach is able to recover the original matrix with a reasonable error compared to the ideal factorization. Our approach is perturbed by the randomness and local optima found by the gradient descent convergence. When compared to other greedy constructive heuristics, we showed that they tend to ignore large parts of secondary components or sometimes tend to overcover unnecessarily.

In chapter 5.1.1, we presented many quantifiable comparisons and our proposed Gantt pipeline for using the Binary Matrix Factorization for process discovery.

First, we compared the time of the BackDisc pipeline using the discretization through a formulation in a modern MIP solver and through our proposed backtracking algorithm. Our proposed backtracking algorithm produced the same factorization within seconds compared to the formulation.

We also compared the error obtained by our BackDisc pipeline (regardless of the discretization methodology) to other algorithms that produce factorizations heuristically. Their error was worse in almost all instances, but interestingly, the factorization error produced by the greedy constructive heuristics worsened with larger ranks G. This effect is a sign that the greedy constructive heuristics cannot manage how much to capture in each component planning to balance the workload between components.

Finally, we obtained a high-level representation of the process for operating the valves in a petrochemical batch process. We segmented one year of operation in two refineries into production cycles and analyzed their similarities. We consequently obtained cohesive time periods in which the processes were similar using the principal component analysis, polynomial filtering, and a heuristic for segmenting time periods. We confirmed with the refinery personnel that the process in refinery 1 usually takes 48 hours, and the process in refinery 2 takes 36 hours.

The selection of cohesive time periods made it possible to obtain segments of the original matrix in which the clusterability (the lower bound error score from reconstructing with the PCA) was lower than the original instance. It also allowed us to run the discretization using the formulation in a MIP solver, which took some minutes even within the selected time period.

The BMF over these time periods did not always produce compelling representations. However, in some cases, it was possible to observe balanced groups of valves and trace parallelism of two different pairs of drums of refinery 1, whose operations are phased to each other to balance the operator's workload. The algorithm could capture this behavior in some of the time periods of refinery 1. We later confirmed this parallel behavior in the real setting.

In chapter 6, we gathered four direct use cases of binary matrix factorization. First, we applied our BackDisc pipeline to datasets used in papers that proposed the heuristic approaches GreConD and TopFiberM. We had to re-run their tests using their code found on the R package rBMF (DESOUKI, 2021) to measure the results using the reconstruction with the regular matrix multiplication. Just as expected, their results had a very high error rate. In the specific instance PAleo, the PCA lower bound was also high, indicating low clusterability; only in this case did GreConD produce a better result than our proposed BackDisc pipeline, even though both error levels were above 80%.

In the following direct applications, we wanted to show some range. We presented three additional domains where Binary Matrix Factorization can be directly used. First, we took inspiration from the following paper from (ZHANG et al., 2010) and apply to gene expression in larger instances than they did. We applied our BackDisc pipeline to 31 datasets from the Gene Expression Omnibus database. Our BackDisc pipeline was able to recover the error loss in the thresholding step with greater success than expected.

In some cases, after discretizing with our BackDisc pipeline, the error was lower than the error with the continuous factorization proposed in (ZHANG et al., 2007). This even lower level of error indicates that the gradient descent approach gets trapped too early in local optima because it is expected for it to perform better. After all, the continuous space contains (is less constrained) than the binary space.

The other two cases had different characteristics, the unbalanced sparseness of columns and rows. In both the topic modeling tests and the recommendation tests, we had to control somehow the sparsity of the problem or test with different sparsities. A too sparse instance often means a sparse relationship graph and nodes that do not belong to clusters. In the topic modeling tests, the sparsity was too high, even working with the 300 most used words, aside from typical stopwords such as prepositions and conjunctions. That means that the use of words in different documents has a great variability.

However, in the topic modeling case, specifically for cases where we used a vocabulary of only 100 words and a G of 50 (in both cases with or without short documents), our BackDisc pipeline could get close to an error level of 60%. This happened because it allowed for capturing pairs of words that determined the topics of texts.

Also, in the recommendation case, we observed the effect of super-users and super-items alongside users and items with very few interactions. These instances have the characteristic of missing data. That means what the user has watched/consumed is not indicative of the totality of what she would have consumed if we gave them time. Thus, it might not encode user preferences in a homogenous manner for all users.

Thus, both in topic modeling and in the recommendation, the PCA lower-bound was high, indicating that they have low clusterability. In many cases, we observed a recovery of at least $20p.p.\%$ of the error. Our best explanation for that is that the sparsity profile is also recovered. The thresholding step typically produces less error with a more sparse choice. Nevertheless, in some cases, the sparsity goes to near zero with the thresholding. In these cases, the sparsity profile is not recovered.

7.2 Strengths and Weaknesses of this approach

Our work's main strength is unifying the accuracy and scalability of continuous methods to the explainability that true binary decompositions have. We were able to demonstrate the possibility of recovering the error that the thresholding step produces.

We also proposed a more robust way of measuring the error of the binary matrix factorization. The reconstruction using regular matrix multiplication allows us to compare the binary matrix factorization to other factorizations, which often are continuous. Also, using the PCA error as a lower bound, we can measure how much loss to expect when compressed to a given rank because different use cases have different natural clusterability.

The main shortcoming of our approach is the dependence on the previous steps of the pipeline prior to our proposed BackDisc discretization. It is tied to the randomness of the alternating least squares gradient descent and its susceptibility to local optima and also to the sparsity profile achieved by thresholding.

Something we have not experienced within our proposed BackDisc pipeline was the control of the sparsity during the thresholding search. Maybe we could generate a better final answer with a worse result in the thresholding procedure.

Furthermore, our approach does not attend to the choice of rank G. Some previous works such as (MIETTINEN; VREEKEN, 2014) discuss a way to find good values that balance the trade-off of choosing too few and not describing well the given matrix or too many producing marginal gains in description or, worse, overlapping components.

7.3 Future Work

As emphasized in the previous section, we presented a constructive heuristic step that post-processes a real-values near binary matrix. The natural ways to expand the work is to add steps to the pipeline or change some premises of the discretization we proposed.

First, we want to test other forms of finding a first approximation to one of the decomposed matrices. We believe that gradient descent is not the only way to do it. We could use some greedy heuristics to accomplish it or from other methods such as K-medoid, random sampling of columns or rows.

Additionally, we believe that we can combine our constructive heuristic with known local search, such as in (MIRISAEE; GAUSSIER; TERMIER, 2015) and (ENE et al., 2008), and test a meta-heuristic search schema to find better solutions, such as in (LU et al., 2011), (SNÃ¡Å¡EL et al., 2007) and (SAENKO; KOTENKO, 2014). Local search algorithms are a promising approach to binary matrix factorization. They are scalable and can be adapted to binary matrix factorization. We can use the results from the gradient descent and subsequent discretization as a starting point for the local search.

Strong heuristic results open the possibility of producing upper bounds for an exact formulation that treats the bilinearity with McCormick envelopes. A research group from Turkey in (KOVACS; GUNLUK; HAUSER, 2021) already proposed a column generation approach to the problem. However, we believe there are still many ways to improve their solution algorithm using tighter bounds, reduced cost fixation, and better heuristic column generation strategies based on our and related works.

Also, we want to adapt our discretization approach by backtracking for the original quadratic formulation or other formulations. We believe that the theoretical step of describing the inclusion gain formula to the linearized case can be adapted to the quadratic formulation or a more general weighted approach that allows the user to tune how to balance between overcovering and undercovering.

8
Bibliography

AALST, W. M. Van der; AALST, W. M. van der. Process discovery: An introduction. **Process mining: Discovery, conformance and enhancement of business processes**, Springer, p. 125–156, 2011.

ASUNCION, A.; NEWMAN, D. **UCI machine learning repository**. [S.l.]: Irvine, CA, USA, 2007.

BARIK, S.; VIKALO, H. Matrix completion and performance guarantees for single individual haplotyping. **IEEE Transactions on Signal Processing 2019**, Institute of Electrical and Electronics Engineers Inc., p. 4782–4794, 2018.

BARRETT, T. et al. Ncbi geo: archive for functional genomics data sets—update. **Nucleic acids research**, Oxford University Press, v. 41, n. D1, p. D991–D995, 2012.

BELOHLAVEK, R.; OUTRATA, J.; TRNECKA, M. Impact of boolean factorization as preprocessing methods for classification of boolean data. **Annals of Mathematics and Artificial Intelligence**, Springer Netherlands, p. 3–22, 2014.

BELOHLAVEK, R.; OUTRATA, J.; TRNECKA, M. Toward quality assessment of boolean matrix factorizations. **Information Sciences**, Elsevier, v. 459, p. 71–85, 2018.

BELOHLAVEK, R.; TRNECKA, M. From-below approximations in boolean matrix factorization: Geometry and new algorithm. **Journal of Computer and System Sciences**, Academic Press Inc., p. 1678–1697, 2013.

BELOHLAVEK, R.; TRNECKA, M. Handling noise in boolean matrix factorization. In: **Proceedings of the 26th International Joint Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 2017. p. 1433–1439.

BELOHLAVEK, R.; VYCHODIL, V. Discovery of optimal factors in binary data via a novel method of matrix decomposition. **Journal of Computer and System Sciences**, Elsevier, v. 76, n. 1, p. 3–20, 2010.

BENNETT, J.; LANNING, S. et al. The netflix prize. In: NEW YORK. **Proceedings of KDD cup and workshop**. [S.l.], 2007. v. 2007, p. 35.

BIRD, S.; KLEIN, E.; LOPER, E. **Natural language processing with Python: analyzing text with the natural language toolkit**. [S.l.]: " O'Reilly Media, Inc.", 2009.

CANTADOR, I.; BRUSILOVSKY, P.; KUFLIK, T. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: **Proceedings of the 5th ACM conference on Recommender systems**. New York, NY, USA: ACM, 2011. (RecSys 2011).

CHANDRAN, U. R. et al. Gene expression profiles of prostate cancer reveal involvement of multiple molecular pathways in the metastatic process. **BMC cancer**, BioMed Central, v. 7, n. 1, p. 1–21, 2007.

CHEMMALAR, S. G.; LAKSHMI, P. G. Rating prediction method for item-based collaborative filtering recommender systems using formal concept analysis. **EAI Endorsed Transactions on Energy Web**, Osterreichischen Gesellschaft fur Politikwissenschaft, p. 1–9, 2021.

CLOUGH, E.; BARRETT, T. The gene expression omnibus database. **Statistical Genomics: Methods and Protocols**, Springer, p. 93–110, 2016.

COMMISSION, A. B. **ABC Corpus**. 2010. Distributed with the Natural Language Toolkit [https://www.nltk.org/nltk$_{d}ata/$].

COMMONS, U. W. **A delayed coker unit**. 2007. File: `Delayed_Coker.jpg`. `Disponível em: <https://upload.wikimedia.org/wikipedia/commons/c/c1/Delayed\_Coker.png>`.

CORRADO, G. et al. Ptrcombiner: mining combinatorial regulation of gene expression from post-transcriptional interaction maps. **BMC Genomics**, BioMed Central Ltd., 2014.

CREIGHTON, C.; HANASH, S. Mining gene expression databases for association rules. **Bioinformatics**, Oxford University Press, v. 19, n. 1, p. 79–86, 2003.

DESOUKI, A. A. **rBMF: Boolean Matrix Factorization**. [S.l.], 2021. R package version 1.1. Disponível em: <https://CRAN.R-project.org/package=rBMF>.

DESOUKI, A. A.; RöDER, M.; NGOMO, A.-C. N. topfiberm: Scalable and efficient boolean matrix factorization. Available at http://arxiv.org/pdf/1903.10326v1 (2019/03/06) | 9 pages, 1 Figure, 3 tables. 2019.

DILLMAN, J. F. et al. Genomic analysis of rodent pulmonary tissue following bis-(2-chloroethyl) sulfide exposure. **Chemical research in toxicology**, ACS Publications, v. 18, n. 1, p. 28–34, 2005.

DIOP, M. et al. Binary matrix factorization applied to netflix dataset analysis. In: **IFAC-PapersOnLine**. [S.l.: s.n.], 2019. p. 13–17.

ECKART, C.; YOUNG, G. The approximation of one matrix by another of lower rank. **Psychometrika**, Springer, v. 1, n. 3, p. 211–218, 1936.

EDGAR, R.; DOMRACHEV, M.; LASH, A. E. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. **Nucleic acids research**, Oxford University Press, v. 30, n. 1, p. 207–210, 2002.

ENE, A. et al. Fast exact and heuristic methods for role minimization problems. In: **Proceedings of the 13th ACM symposium on Access control models and technologies**. [S.l.: s.n.], 2008. p. 1–10.

FRANCIS, W. N.; KUCERA, H. Brown corpus manual. **Letters to the Editor**, v. 5, n. 2, p. 7, 1979.

FRANK, M. et al. Mining permission request patterns from android and facebook applications (extended author version). 2012.

GOLDEN, J.; O'MALLEY, D. Reverse annealing for nonnegative/binary matrix factorization. **Plos one**, Public Library of Science San Francisco, CA USA, v. 16, n. 1, p. e0244026, 2021.

GOLUB, G. H.; LOAN, C. F. V. **Matrix computations**. 4. ed. [S.l.]: JHU Press, 2012.

GRIFFITHS, T. L.; STEYVERS, M. Finding scientific topics. **Proceedings of the National academy of Sciences**, National Acad Sciences, v. 101, n. suppl_1, p. 5228–5235, 2004.

GROSMAN, J. **Findpapers: A tool for helping researchers who are looking for related works**. 2020. <https://github.com/jonatasgrosman/findpapers>.

Gurobi Optimization, LLC. **Gurobi Optimizer Reference Manual**. 2023. Disponível em: <https://www.gurobi.com>.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. **Acm transactions on interactive intelligent systems (tiis)**, Acm New York, NY, USA, v. 5, n. 4, p. 1–19, 2015.

HART, M. Project gutenberg mission statement. **Project Gutenberg**, 2004.

HESS, S.; MORIK, K.; PIATKOWSKI, N. The primping routine—tiling through proximal alternating linearized minimization. **Data Mining and Knowledge Discovery**, Springer Netherlands, p. 1090–1131, 2017.

HINZE, C. H. et al. Immature cell populations and an erythropoiesis gene-expression signature in systemic juvenile idiopathic arthritis: implications for pathogenesis. **Arthritis research & therapy**, Springer, v. 12, p. 1–13, 2010.

IWAKAWA, M. et al. The radiation-induced cell-death signaling pathway is activated by concurrent use of cisplatin in sequential biopsy specimens from patients with cervical cancer. **Cancer biology & therapy**, Taylor & Francis, v. 6, n. 6, p. 905–911, 2007.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. **Computer**, IEEE, v. 42, n. 8, p. 30–37, 2009.

KOVACS, R.; GUNLUK, O.; HAUSER, R. Low-rank boolean matrix approximation by integer programming. **arXiv preprint arXiv:1803.04825**, 2018.

KOVACS, R. A.; GUNLUK, O.; HAUSER, R. A. Binary matrix factorisation via column generation. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2021. v. 35, n. 5, p. 3823–3831.

KUMAR, R. et al. Faster algorithms for binary matrix factorization. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2019. p. 3551–3559.

LABRECHE, H. G.; NEVINS, J. R.; HUANG, E. Integrating factor analysis and a transgenic mouse model to reveal a peripheral blood predictor of breast tumors. **BMC medical genomics**, BioMed Central, v. 4, n. 1, p. 1–14, 2011.

LANGE, H.; BERGéS, M. Bolt: Energy disaggregation by online binary matrix factorization of current waveforms. In: **Proceedings of the 3rd ACM Conference on Systems for Energy-Efficient Built Environments, BuildSys 2016**. [S.l.]: Association for Computing Machinery, 2016. p. 11–20.

LECUN, Y. The mnist database of handwritten digits. **http://yann. lecun. com/exdb/mnist/**, 1998.

LEE, D. D.; SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. **Nature**, Nature Publishing Group UK London, v. 401, n. 6755, p. 788–791, 1999.

LI, F. et al. Detection and identification of cyber and physical attacks on distribution power grids with pvs: An online high-dimensional data-driven approach. **IEEE Journal of Emerging and Selected Topics in Power Electronics**, IEEE, v. 10, n. 1, p. 1282–1291, 2019.

LI, F. et al. Dete ction and identification of cyber and physical attacks on distribution power grids with pvs: An online high-dimensional data-driven approach. **IEEE Journal of Emerging and Selected Topics in Power Electronics**, Institute of Electrical and Electronics Engineers Inc., p. 1282–1291, 2022.

LIU, Y.-C.; CHENG, C.-P.; TSENG, V. S. Mining differential top-k co-expression patterns from time course comparative gene expression datasets. **BMC bioinformatics**, Springer, v. 14, p. 1–13, 2013.

LU, H. et al. Weighted rank-one binary matrix factorization. In: **Proceedings of the 11th SIAM International Conference on Data Mining, SDM 2011**. [S.l.: s.n.], 2011. p. 283–294.

LUCCHESE, C.; ORLANDO, S.; PEREGO, R. Mining top-k patterns from binary datasets in presence of noise. In: SIAM. **Proceedings of the 2010 SIAM International Conference on Data Mining**. [S.l.], 2010. p. 165–176.

LUCCHESE, C.; ORLANDO, S.; PEREGO, R. A unifying framework for mining approximate top-k binary patterns. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 26, n. 12, p. 2900–2913, 2013.

MALIK, O. A. et al. Binary matrix factorization on special purpose hardware. **PLOS ONE 16(12): e0261250, 2021**, Public Library of Science, p. e0261250, 2021.

MCCORMICK, G. P. Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. **Mathematical programming**, Springer, v. 10, n. 1, p. 147–175, 1976.

MEEDS, E. et al. Modeling dyadic data with binary latent factors. **Advances in neural information processing systems**, v. 19, 2006.

METZELER, K. H. et al. An 86-probe-set gene-expression signature predicts survival in cytogenetically normal acute myeloid leukemia. **Blood, The Journal of the American Society of Hematology**, American Society of Hematology Washington, DC, v. 112, n. 10, p. 4193–4201, 2008.

MEYER, L. H. et al. Early relapse in all is identified by time to leukemia in nod/scid mice and is characterized by a gene signature involving survival pathways. **Cancer cell**, Elsevier, v. 19, n. 2, p. 206–217, 2011.

MIETTINEN, P. Matrix decomposition methods for data mining: Computational complexity and algorithms. Helsingin yliopisto, 2009.

MIETTINEN, P. et al. The discrete basis problem. **IEEE transactions on knowledge and data engineering**, IEEE, v. 20, n. 10, p. 1348–1362, 2008.

MIETTINEN, P.; NEUMANN, S. Recent developments in boolean matrix factorization. In: **Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2021. p. 4922–4928.

MIETTINEN, P.; VREEKEN, J. Mdl4bmf: Minimum description length for boolean matrix factorization. **ACM Transactions on Knowledge Discovery from Data**, Association for Computing Machinery (ACM), p. 1–31, 2014.

MIRISAEE, H.; GAUSSIER, E.; TERMIER, A. Efficient local search for l1 and l2 binary matrix factorization. **Intelligent Data Analysis**, IOS Press BV, p. 783–807, 2016.

MIRISAEE, S. H.; GAUSSIER, E.; TERMIER, A. Improved local search for binary matrix factorization. In: **Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 2015. p. 1198–1204.

MONKS, S. et al. Genetic inheritance of gene expression in human cell lines. **The American Journal of Human Genetics**, Elsevier, v. 75, n. 6, p. 1094–1105, 2004.

MORSE, C. G. et al. Hiv infection and antiretroviral therapy have divergent effects on mitochondria in adipose tissue. **Journal of Infectious Diseases**, Oxford University Press, v. 205, n. 12, p. 1778–1787, 2012.

NAIR, R. P. et al. Genome-wide scan reveals association of psoriasis with il-23 and nf-κb pathways. **Nature genetics**, Nature Publishing Group US New York, v. 41, n. 2, p. 199–204, 2009.

NOBLE, C. L. et al. Regional variation in gene expression in the healthy colon is dysregulated in ulcerative colitis. **Gut**, BMJ Publishing Group, v. 57, n. 10, p. 1398–1405, 2008.

PANG, B.; LEE, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: **Proceedings of the ACL**. [S.l.: s.n.], 2004.

PARFENOV, D. et al. Research of genetic optimization algorithms in the design of vlan. In: **2021 29th Telecommunications Forum, TELFOR 2021 - Proceedings**. [S.l.: s.n.], 2021.

PARIKH, N.; BOYD, S. et al. Proximal algorithms. **Foundations and trends® in Optimization**, Now Publishers, Inc., v. 1, n. 3, p. 127–239, 2014.

PARNELL, G. P. et al. Identifying key regulatory genes in the whole blood of septic patients to monitor underlying immune dysfunctions. **Shock**, LWW, v. 40, n. 3, p. 166–174, 2013.

PELLAGATTI, A. et al. Deregulated gene expression pathways in myelodysplastic syndrome hematopoietic stem cells. **Leukemia**, Nature Publishing Group, v. 24, n. 4, p. 756–764, 2010.

PFISTER, T. D. et al. Topoisomerase i levels in the nci-60 cancer cell line panel determined by validated elisa and microarray analysis and correlation with indenoisoquinoline sensitivity. **Molecular cancer therapeutics**, AACR, v. 8, n. 7, p. 1878–1884, 2009.

PRAT, A. et al. based pam50 subtype predictor identifies higher responses and improved survival outcomes in her2-positive breast cancer in the noah study. **Clinical Cancer Research**, AACR, v. 20, n. 2, p. 511–521, 2014.

PRESIDENTS, V. U. **Inaugural Address Corpus**. 1789–2009. Distributed with the Natural Language Toolkit [https://www.nltk.org/nltk$_data/$].

RAVANBAKHSH, S.; POCZOS, B.; GREINER, R. Boolean matrix factorization and noisy completion via message passing. Available at http://arxiv.org/pdf/1509.08535v3 (2015/09/28). 2015.

RIJSBERGEN, C. J. V.; ROBERTSON, S. E.; PORTER, M. F. **New models in probabilistic information retrieval**. [S.l.]: British Library Research and Development Department London, 1980. v. 5587.

RUKAT, T. et al. Bayesian boolean matrix factorisation. In: **Proceedings of the 34th International Conference on Machine Learning - Volume 70**. [S.l.]: JMLR.org, 2017. p. 2969–2978.

SAENKO, I.; KOTENKO, I. Design of virtual local area network scheme based on genetic optimization and visual analysis. **Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications**, Innovative Information Science Technology Research Group (ISYOU), p. 86–102, 2014.

SAENKO, I.; KOTENKO, I. **A Genetic Approach for Virtual Computer Network Design**. [S.l.]: Springer Verlag, 2015. 95-105 p.

SAENKO, I.; KOTENKO, I. **Reconfiguration of Access Schemes in Virtual Networks of the Internet of Things by Genetic Algorithms**. [S.l.]: Springer Verlag, 2016. 155-165 p.

SAVITZKY, A.; GOLAY, M. J. Smoothing and differentiation of data by simplified least squares procedures. **Analytical chemistry**, ACS Publications, v. 36, n. 8, p. 1627–1639, 1964.

SAWARKAR, A. N. et al. Petroleum residue upgrading via delayed coking: A review. **The Canadian Journal of Chemical Engineering**, Wiley Online Library, v. 85, n. 1, p. 1–24, 2007.

SCHENA, M. et al. Quantitative monitoring of gene expression patterns with a complementary dna microarray. **Science**, American Association for the Advancement of Science, v. 270, n. 5235, p. 467–470, 1995.

SCHIRMER, S. H. et al. Suppression of inflammatory signaling in monocytes from patients with coronary artery disease. **Journal of molecular and cellular cardiology**, Elsevier, v. 46, n. 2, p. 177–185, 2009.

SHAPIRA, S. D. et al. A physical and regulatory map of host-influenza interactions reveals pathways in h1n1 infection. **Cell**, Elsevier, v. 139, n. 7, p. 1255–1267, 2009.

SHEN, B.-H.; JI, S.; YE, J. Mining discrete patterns via binary matrix factorization. In: **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.: s.n.], 2009. p. 757–765.

SNÃ¡Å¡EL, V. et al. On the road to genetic boolean matrix factorization. **Neural Network World**, Czech Technical University in Prague, p. 675–688, 2007.

SPIRA, A. et al. Airway epithelial gene expression in the diagnostic evaluation of smokers with suspect lung cancer. **Nature medicine**, Nature Publishing Group US New York, v. 13, n. 3, p. 361–366, 2007.

STEAM Video Games. <https://www.kaggle.com/datasets/tamber/steam-video-games>. Accessed: 2022-10-26.

SU, A. I. et al. A gene atlas of the mouse and human protein-encoding transcriptomes. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 101, n. 16, p. 6062–6067, 2004.

SUÁREZ-FARINAS, M. et al. Expanding the psoriasis disease profile: interrogation of the skin and serum of patients with moderate-to-severe psoriasis. **Journal of Investigative Dermatology**, Elsevier, v. 132, n. 11, p. 2552–2564, 2012.

SUN, L. et al. Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain. **Cancer cell**, Elsevier, v. 9, n. 4, p. 287–300, 2006.

TASKESEN, E. et al. Prognostic impact, concurrent genetic mutations, and gene expression features of aml with cebpa mutations in a cohort of 1182 cytogenetically normal aml patients: further evidence for cebpa double mutant aml as a distinctive disease entity. **Blood, The Journal of the American Society of Hematology**, American Society of Hematology Washington, DC, v. 117, n. 8, p. 2469–2475, 2011.

TIAN, E. et al. The role of the wnt-signaling antagonist dkk1 in the development of osteolytic lesions in multiple myeloma. **New England Journal of Medicine**, Mass Medical Soc, v. 349, n. 26, p. 2483–2494, 2003.

TU, S.; CHEN, R.; XU, L. A binary matrix factorization algorithm for protein complex prediction. **Proteome science**, BioMed Central Ltd., 2011.

VARIOUS. **Webtext Corpus**. 2005. Distributed with the Natural Language Toolkit [https://www.nltk.org/nltk$_{d}ata/$].

VOTAVOVA, H. et al. Transcriptome alterations in maternal and fetal cells induced by tobacco smoke. **Placenta**, Elsevier, v. 32, n. 10, p. 763–770, 2011.

WAN, C. et al. Fast and efficient boolean matrix factorization by geometric segmentation. Available at http://arxiv.org/abs/1909.03991v2 (2019/09/09) | Accepted at AAAI 2020. 2019.

WANG, J. et al. **Linear Policy Recommender Scheme for Large-Scale Attribute-Based Access Control**. [S.l.]: Springer Verlag, 2022. 175-191 p.

WILLIAMS, K.; CALVO, R. A. **A framework for text categorization**. Tese (Doutorado) — Citeseer, 2003.

WOLD, S.; ESBENSEN, K.; GELADI, P. Principal component analysis. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 2, n. 1-3, p. 37–52, 1987.

YOGISH, D.; MANJUNATH, T.; HEGADI, R. S. Review on natural language processing trends and techniques using nltk. In: SPRINGER. **Recent Trends in Image Processing and Pattern Recognition: Second International Conference, RTIP2R 2018, Solapur, India, December 21–22, 2018, Revised Selected Papers, Part III 2**. [S.l.], 2019. p. 589–606.

ZHANG, Z. et al. Binary matrix factorization with applications. In: IEEE. **Seventh IEEE international conference on data mining (ICDM 2007)**. [S.l.], 2007. p. 391–400.

ZHANG, Z.-Y.; AHN, Y.-Y. Community detection in bipartite networks using weighted symmetric binary matrix factorization. **International Journal of Modern Physics C**, World Scientific Publishing Co. Pte Ltd, 2015.

ZHANG, Z.-Y. et al. Binary matrix factorization for analyzing gene expression data. **Data Mining and Knowledge Discovery**, Springer, v. 20, p. 28–52, 2010.

ZHANG, Z.-Y.; WANG, Y.; AHN, Y.-Y. Overlapping community detection in complex networks using symmetric binary matrix factorization. **Physical Review E**, APS, v. 87, n. 6, p. 062803, 2013.

ZITNIK, M.; ZUPAN, B. Nimfa: A python library for nonnegative matrix factorization. **Journal of Machine Learning Research**, v. 13, p. 849–853, 2012.

A
Gantt Charts

This appendix shows all the Gantt charts generated by the process dicovery pipeline presented in chapter 5.1.1.

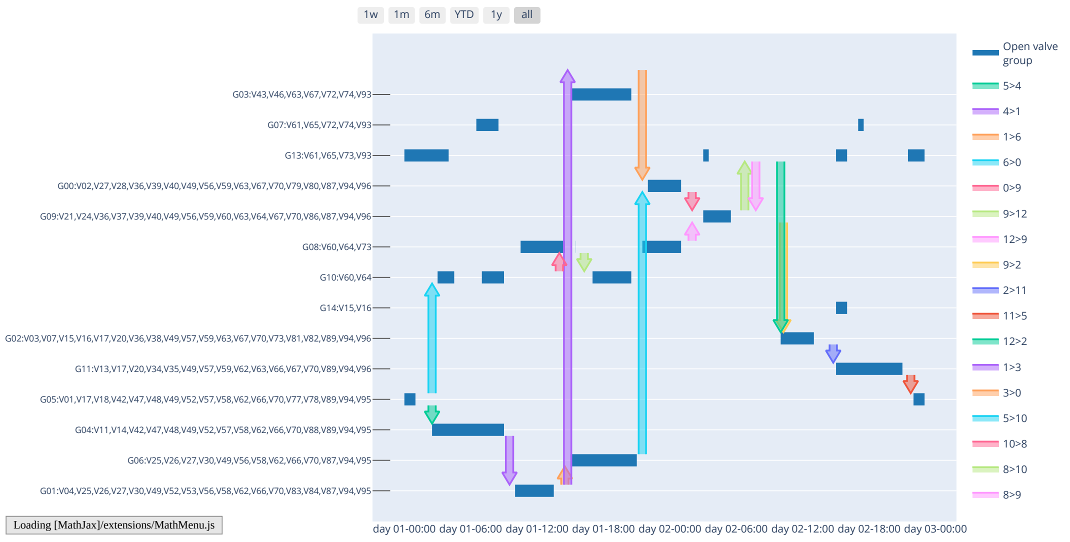


Figure A.1: Gantt chart of the first cohesive time period selected of the first refinery, comprehending 9 production cycles.

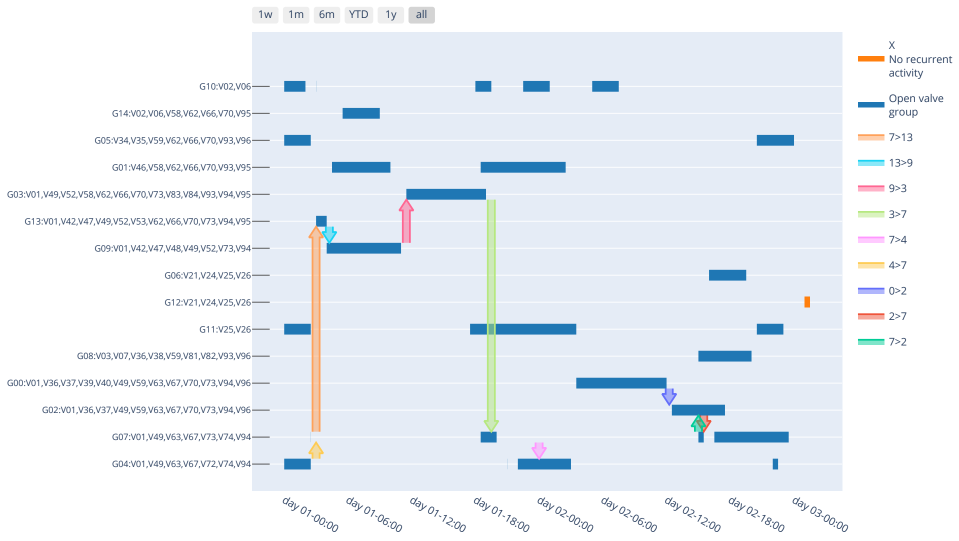


Figure A.2: Gantt chart of the second cohesive time period selected of the first refinery, comprehending 7 production cycles.

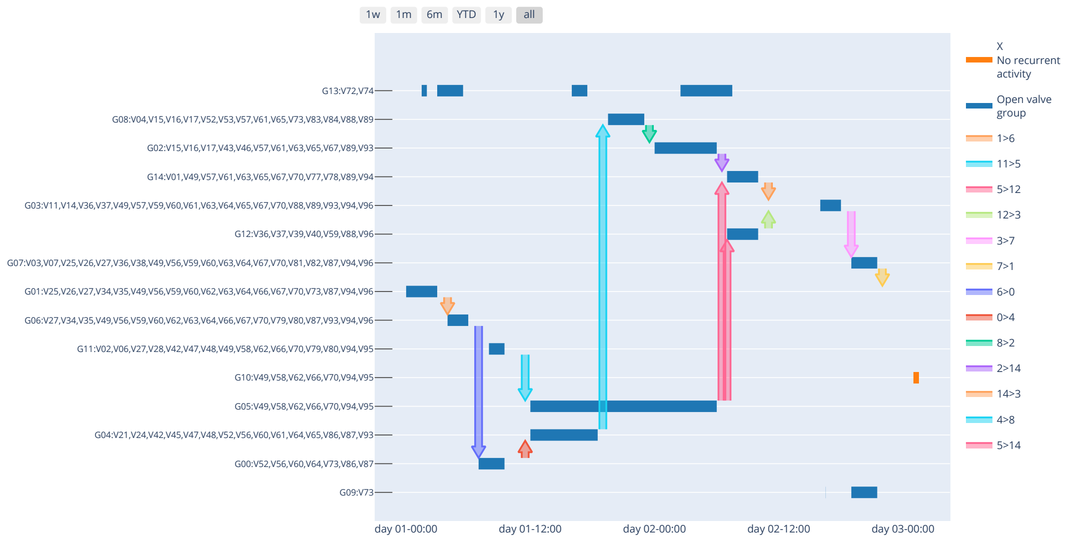


Figure A.3: Gantt chart of the third cohesive time period selected of the first refinery, comprehending 6 production cycles.

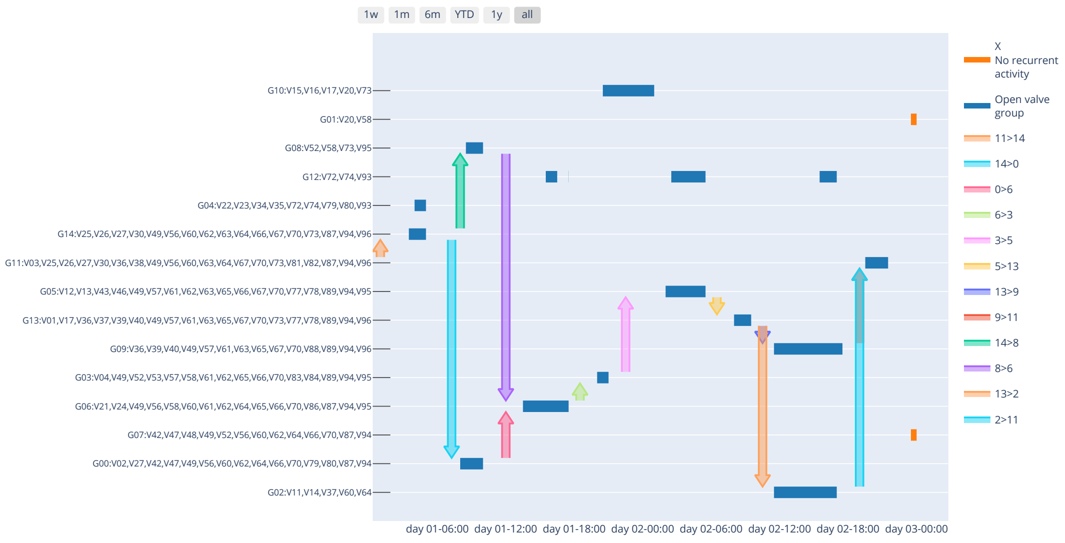


Figure A.4: Gantt chart of the fourth cohesive time period selected of the first refinery, comprehending 6 production cycles.

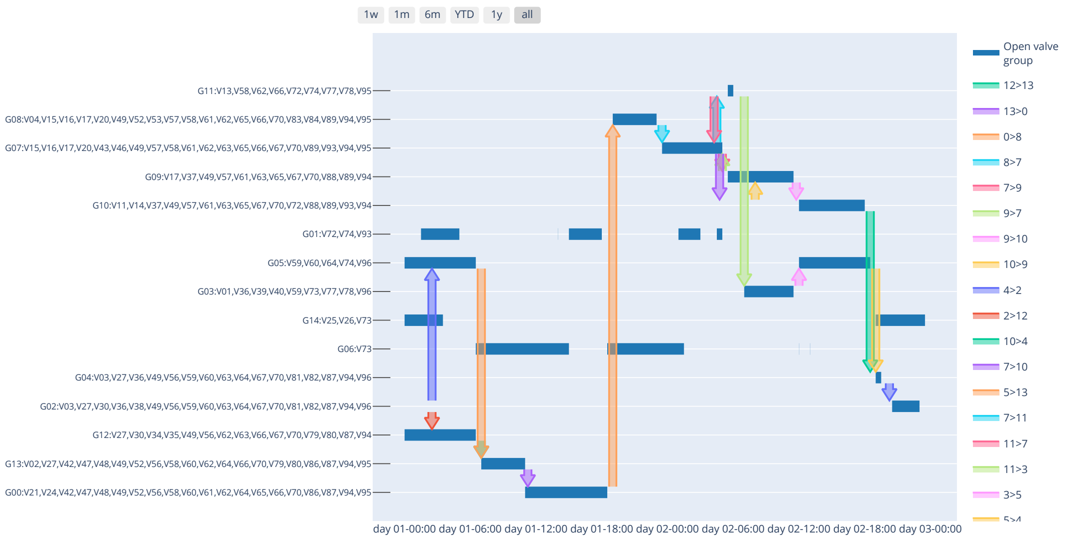


Figure A.5: Gantt chart of the fifth cohesive time period selected of the first refinery, comprehending 15 production cycles.

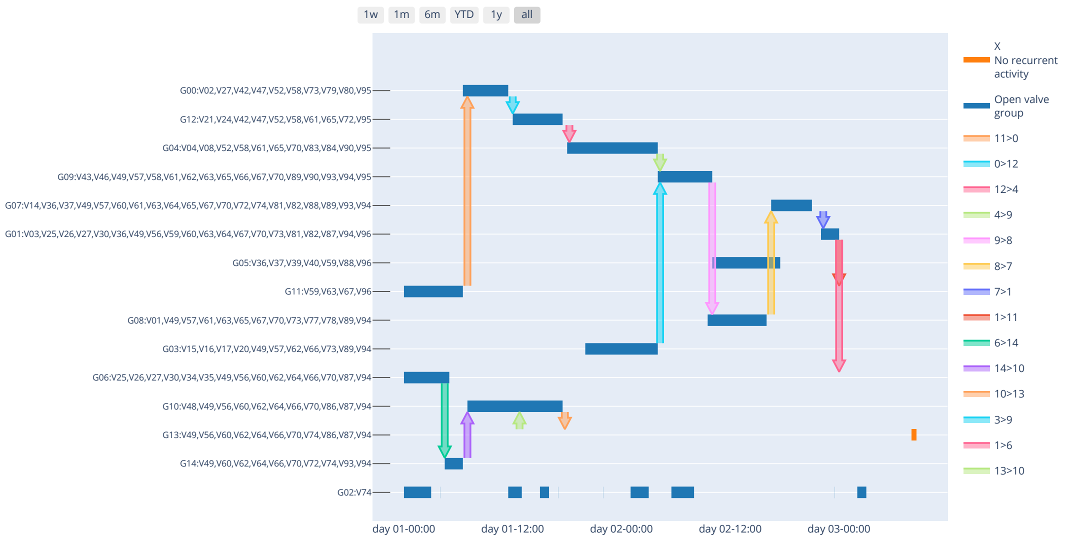


Figure A.6: Gantt chart of the sixth cohesive time period selected of the first refinery, comprehending 7 production cycles.

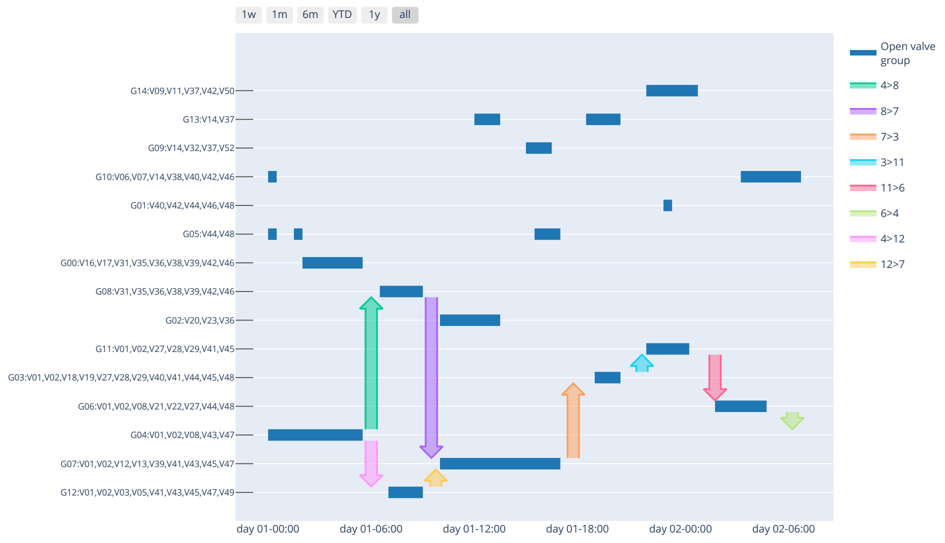


Figure A.7: Gantt chart of the first cohesive time period selected of the second refinery, comprehending 8 production cycles.

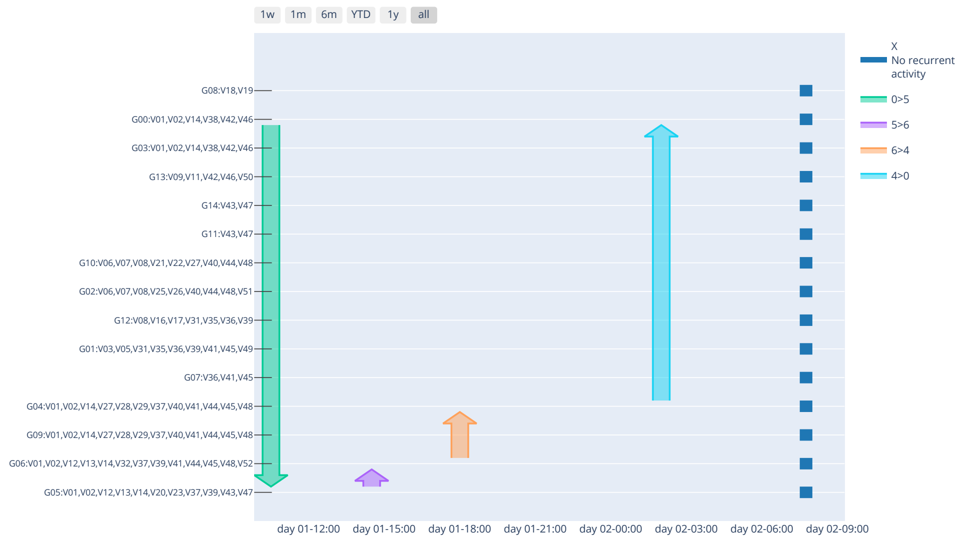


Figure A.8: Gantt chart of the second cohesive time period selected of the second refinery, comprehending 13 production cycles.

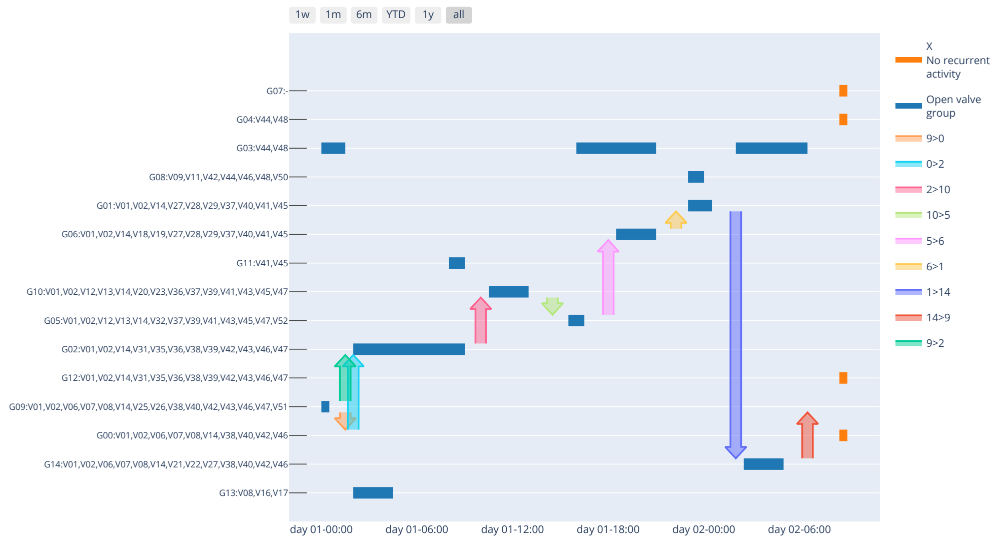


Figure A.9: Gantt chart of the third cohesive time period selected of the second refinery, comprehending 23 production cycles.

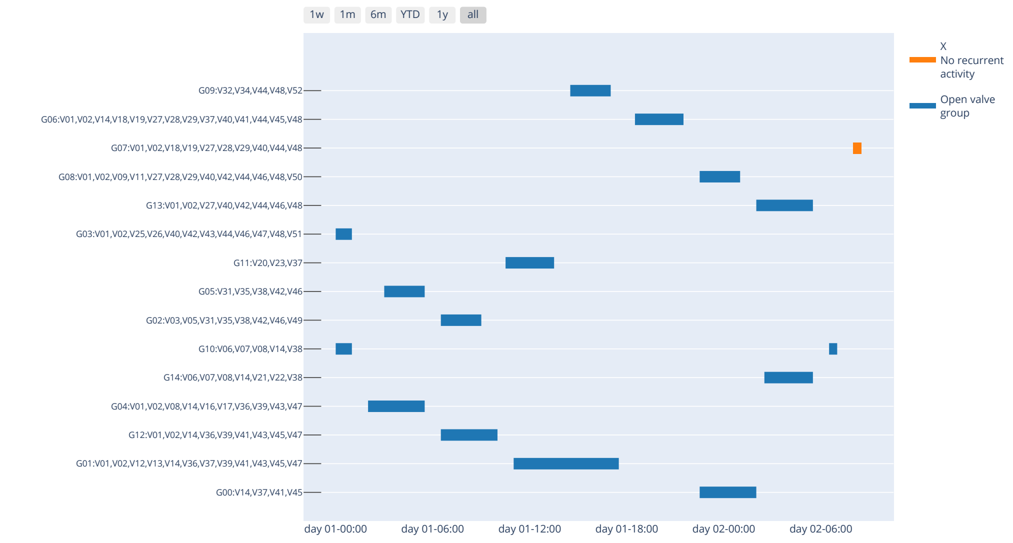


Figure A.10: Gantt chart of the fourth cohesive time period selected of the second refinery, comprehending 40 production cycles.

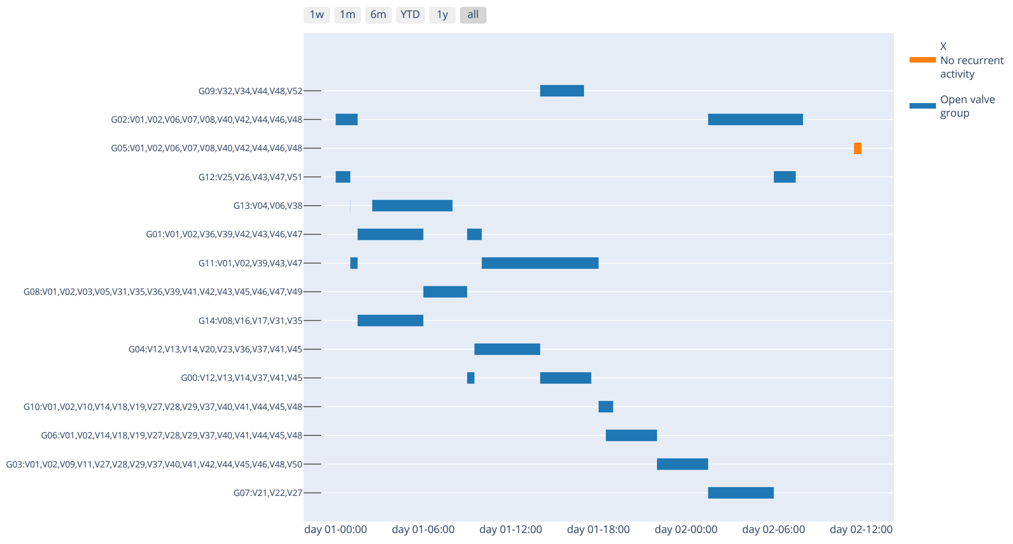


Figure A.11: Gantt chart of the fifth cohesive time period selected of the second refinery, comprehending 47 production cycles.

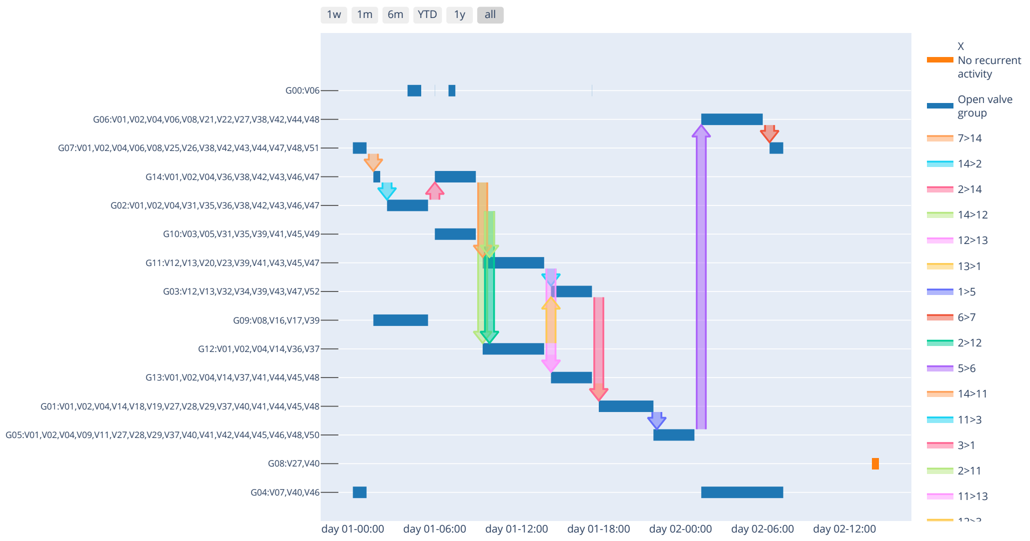


Figure A.12: Gantt chart of the sixth cohesive time period selected of the second refinery, comprehending 20 production cycles.

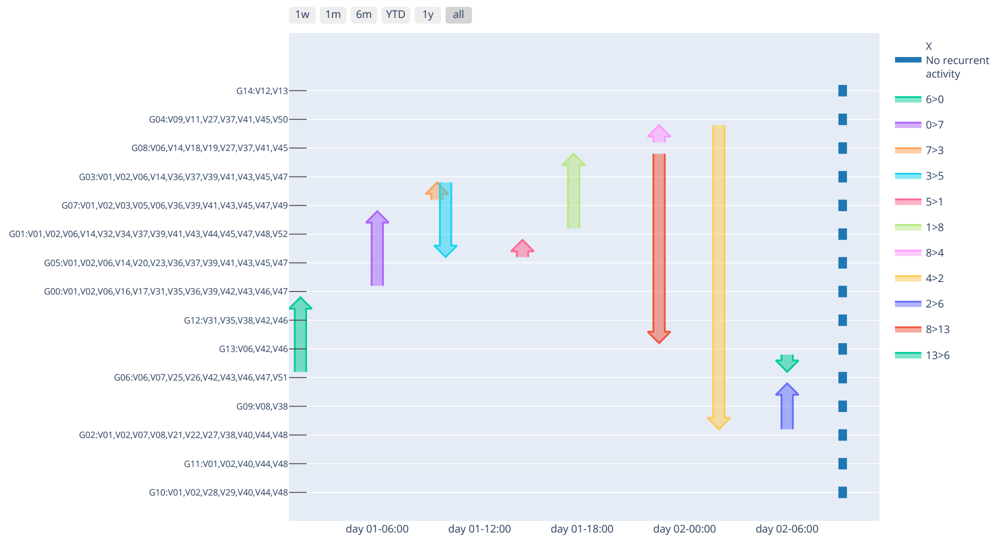


Figure A.13: Gantt chart of the seventh cohesive time period selected of the second refinery, comprehending 6 production cycles.

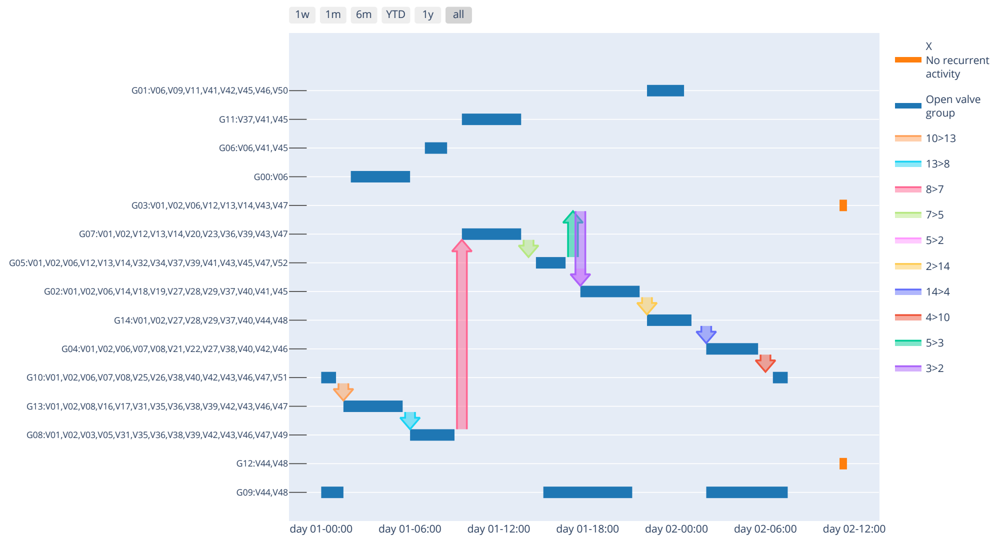


Figure A.14: Gantt chart of the eighth cohesive time period selected of the second refinery, comprehending 28 production cycles.

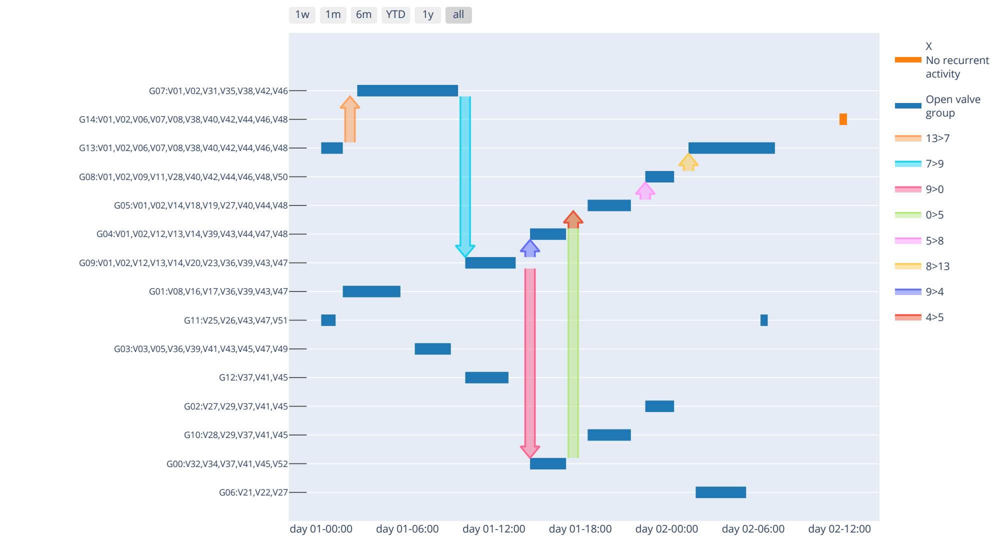


Figure A.15: Gantt chart of the ninth cohesive time period selected of the second refinery, comprehending 15 production cycles.

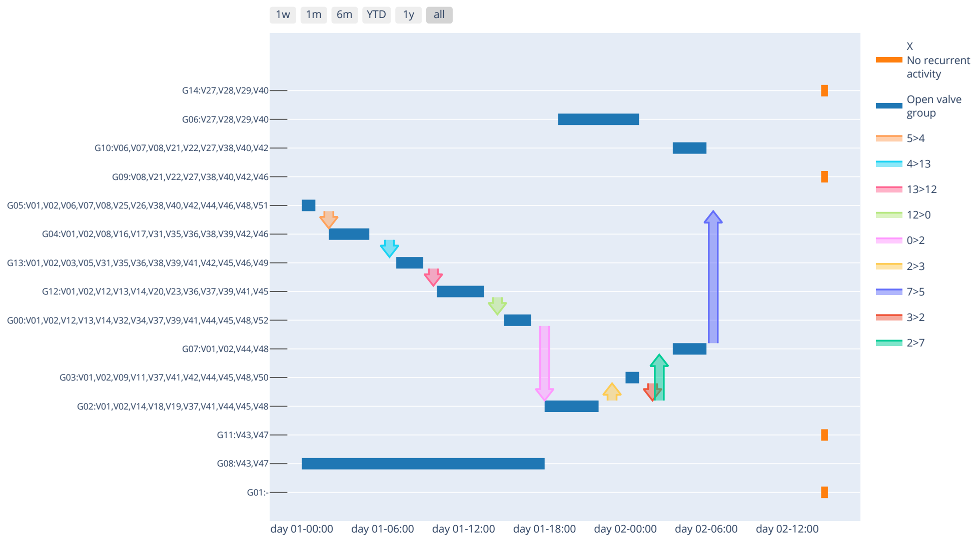


Figure A.16: Gantt chart of the tenth cohesive time period selected of the second refinery, comprehending 11 production cycles.

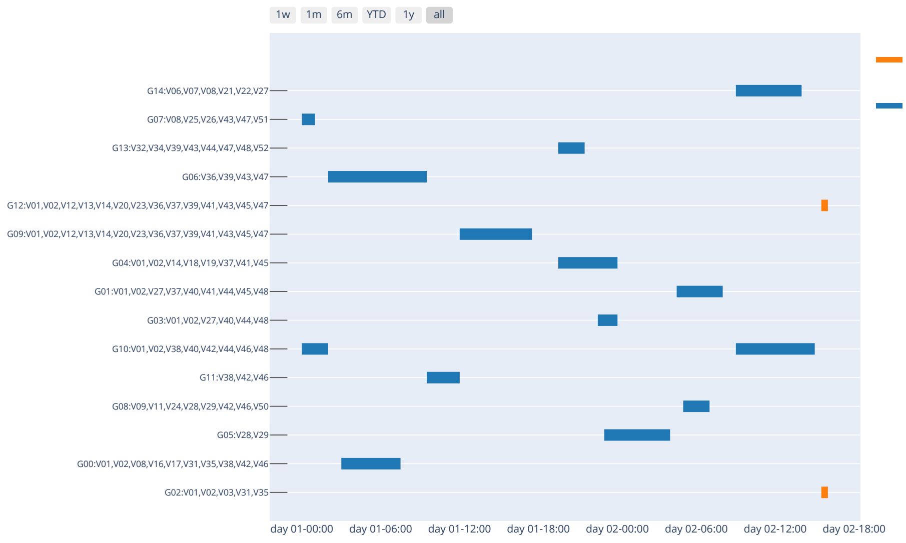


Figure A.17: Gantt chart of the eleventh cohesive time period selected of the second refinery, comprehending 21 production cycles.

B
Detailed Result Tables

This appendix has the complementary tables for the Gene Expression and Natural Language Processing applications presented in chapter 6.

Original					PCA	Zhang		Thresholding			BackDisc			
GEO matrix	rows (#)	cols (#)	Sp (%)	G	Err (%)	time (s)	Err (%)	time (s)	Spars (%)	Err (%)	W time (s)	H time (s)	Spars (%)	Err (%)
GSE11223	40991	404	24	10	31	64	59	8	6	93	150	3	20	65
GSE11223	40991	404	24	20	27	98	54	6	19	95	169	4	20	62
GSE11223	40991	404	24	50	22	306	50	10	13	85	1780	25	21	59
GSE1133-GPL1073	31373	244	24	10	30	30	58	3	5	96	58	1	21	65
GSE1133-GPL1073	31373	244	24	20	27	55	53	3	3	96	58	2	23	64
GSE1133-GPL1073	31373	244	24	50	20	181	48	5	12	83	1402	11	22	56
GSE1133-GPL1074	11391	316	24	10	43	13	64	1	4	96	10	0	19	76
GSE1133-GPL1074	11391	316	24	20	40	24	61	1	2	97	13	0	22	77
GSE1133-GPL1074	11391	316	24	50	34	73	57	2	9	90	132	6	20	70
GSE1133-GPL96	22283	316	24	10	40	27	65	3	7	93	37	1	19	73
GSE1133-GPL96	22283	316	24	20	37	48	60	2	3	95	41	3	20	73
GSE1133-GPL96	22283	316	24	50	30	145	55	4	13	86	1075	10	21	67
GSE12417-GPL570	54675	158	24	10	16	36	45	4	4	95	112	1	24	48
GSE12417-GPL570	54675	158	24	20	13	69	39	3	19	98	127	1	24	45
GSE12417-GPL570	54675	158	24	50	9	239	35	4	7	88	194	10	24	43
GSE12417-GPL96	22283	326	24	10	16	29	46	3	6	92	39	1	23	47
GSE12417-GPL96	22283	326	24	20	14	49	41	2	18	95	53	1	23	46
GSE12417-GPL96	22283	326	24	50	11	141	36	3	17	89	973	4	23	45
GSE12417-GPL97	22477	326	24	10	15	24	45	2	5	94	37	1	22	48
GSE12417-GPL97	22477	326	24	20	13	45	40	2	14	93	50	2	23	45
GSE12417-GPL97	22477	326	24	50	11	133	36	3	16	91	933	9	23	44
GSE13355	54675	360	25	10	63	92	77	11	6	97	241	2	12	90
GSE13355	54675	360	25	20	57	192	74	15	9	94	246	2	12	89
GSE13355	54675	360	25	50	48	411	70	15	7	93	381	9	13	87
GSE13576	54675	418	24	10	18	76	48	10	4	97	277	3	23	52
GSE13576	54675	418	24	20	17	146	43	9	16	94	314	6	24	48
GSE13576	54675	418	24	50	14	430	39	12	18	92	2526	30	23	48
GSE1726	23880	334	24	10	57	39	76	4	8	96	46	0	13	89
GSE1726	23880	334	24	20	51	58	72	3	5	96	48	1	14	87
GSE1726	23880	334	24	50	41	167	67	6	11	91	79	2	15	84
GSE1726	23880	334	24	100	31	350	62	6	11	89	2984	10	17	80
GSE1888	15923	308	24	10	15	18	44	1	4	95	19	0	24	45
GSE1888	15923	308	24	20	13	32	38	1	16	95	27	1	24	43
GSE1888	15923	308	24	50	10	96	35	2	15	90	394	3	23	43
GSE19392	22277	338	24	10	13	25	41	2	5	94	38	1	24	43
GSE19392	22277	338	24	20	12	44	36	2	13	94	52	1	24	40
GSE19392	22277	338	24	50	9	129	33	2	9	90	202	7	24	39
GSE19392	22277	338	24	100	6	285	31	2	6	92	535	25	24	40
GSE19429	54675	400	25	10	14	66	44	9	3	96	251	2	23	45
GSE19429	54675	400	25	20	13	127	38	7	12	93	280	4	24	43
GSE19429	54675	400	25	50	11	380	35	9	12	89	2608	21	24	43
GSE21521	54675	308	25	10	72	87	81	11	5	98	248	1	7	95
GSE21521	54675	308	25	20	66	169	78	13	7	97	208	1	8	94
GSE21521	54675	308	25	50	55	374	74	15	6	95	237	3	10	91
GSE21521	54675	308	25	100	41	784	70	15	9	91	398	12	13	82
GSE22845	54675	308	25	10	78	88	81	9	0	100	205	0	0	100
GSE22845	54675	308	25	20	73	135	79	8	1	99	212	1	1	97
GSE22845	54675	308	25	50	61	366	76	12	4	97	234	4	9	93
GSE22845	54675	308	25	100	44	793	72	15	9	92	355	12	10	84
GSE27272	24526	366	24	10	23	60	55	7	11	85	56	0	23	56
GSE27272	24526	366	24	20	22	60	48	3	3	96	55	1	23	57
GSE27272	24526	366	24	50	17	182	44	4	8	90	172	8	24	52

Original					PCA	Zhang		Thresholding			BackDisc			
GEO matrix	rows	cols	Sp	G	Err	time	Err	time	Spars	Err	W time	H time	Spars	Err
	(#)	(#)	(%)		(%)	(s)	(%)	(s)	(%)	(%)	(s)	(s)	(%)	(%)
GSE27567-GPL1261	45101	186	24	10	11	31	41	3	6	91	88	1	24	41
GSE27567-GPL1261	45101	186	24	20	10	60	35	2	17	90	105	1	24	38
GSE27567-GPL1261	45101	186	24	50	6	209	31	4	18	93	1150	4	24	38
GSE27567-GPL570	54675	324	24	10	12	59	41	8	4	96	220	2	24	42
GSE27567-GPL570	54675	324	24	20	11	113	35	5	13	89	241	3	24	40
GSE27567-GPL570	54675	324	24	50	9	353	32	9	13	86	2308	21	23	39
GSE30310	4776	332	24	10	73	9	82	0	6	99	3	0	6	96
GSE30310	4776	332	24	20	64	15	78	1	6	97	3	0	8	94
GSE30310	4776	332	24	50	51	36	74	1	7	94	6	0	11	88
GSE30310	4776	332	24	100	37	73	69	1	10	90	101	1	13	82
GSE30999	54675	340	24	10	7	75	38	9	5	93	277	2	24	32
GSE30999	54675	340	24	20	6	108	30	6	12	88	337	6	24	31
GSE30999	54675	340	24	50	4	426	26	7	10	91	639	24	24	30
GSE30999	54675	340	24	100	3	906	24	11	5	92	1788	25	24	31
GSE32474	54675	348	24	10	17	65	47	8	5	96	233	2	24	48
GSE32474	54675	348	24	20	15	121	41	7	15	93	284	5	24	46
GSE32474	54675	348	24	50	11	378	38	9	16	89	2817	15	23	45
GSE3578	54359	312	24	10	16	55	46	8	3	97	197	2	22	52
GSE3578	54359	312	24	20	14	104	40	6	14	95	230	3	24	45
GSE3578	54359	312	24	50	10	334	36	9	18	89	2050	12	24	43
GSE4115	22215	384	25	10	13	34	45	3	5	92	47	1	23	43
GSE4115	22215	384	25	20	11	54	38	3	18	91	64	2	24	42
GSE4115	22215	384	25	50	9	153	34	3	17	92	695	6	24	40
GSE4290	54613	360	24	10	23	70	51	9	4	96	242	2	23	56
GSE4290	54613	360	24	20	21	128	47	7	18	97	283	3	23	54
GSE4290	54613	360	24	50	18	391	43	9	5	93	356	25	24	53
GSE50948	54675	312	24	10	22	62	49	7	5	95	211	2	23	54
GSE50948	54675	312	24	20	21	114	44	6	16	96	241	3	23	51
GSE50948	54675	312	24	50	18	327	42	7	8	93	601	17	23	52
GSE54514	24840	326	25	10	33	34	59	3	7	93	48	1	22	66
GSE54514	24840	326	25	20	30	57	55	3	2	97	49	2	22	68
GSE54514	24840	326	25	50	24	164	50	4	8	92	135	9	22	64
GSE6919-GPL8300	12625	342	24	10	31	16	56	1	5	96	14	0	21	65
GSE6919-GPL8300	12625	342	24	20	29	28	52	1	0	99	13	0	24	71
GSE6919-GPL8300	12625	342	24	50	24	84	49	2	5	94	53	5	23	61
GSE6919-GPL92	12620	336	24	10	38	15	61	1	4	96	13	0	22	71
GSE6919-GPL92	12620	336	24	20	36	30	57	1	1	98	14	0	20	74
GSE6919-GPL92	12620	336	24	50	30	83	54	2	5	94	67	6	22	67
GSE6919-GPL93	12646	330	24	10	40	17	62	1	6	95	15	0	21	70
GSE6919-GPL93	12646	330	24	20	38	28	58	1	0	99	13	0	24	77
GSE6919-GPL93	12646	330	24	50	32	83	55	2	3	96	25	4	21	71
GSE755	12625	346	24	10	28	16	53	1	4	97	15	0	22	61
GSE755	12625	346	24	20	26	29	49	1	14	97	21	1	23	58
GSE755	12625	346	24	50	21	86	46	1	16	91	600	6	22	59
GSE9820	20589	306	24	10	16	24	51	2	9	87	30	1	23	51
GSE9820	20589	306	24	20	14	42	43	2	9	85	34	2	24	43
GSE9820	20589	306	24	50	11	121	37	3	15	79	591	4	24	41
GSE9820	20589	306	24	100	7	265	34	3	11	84	2570	15	24	40

	Original					PCA	Zhang		Thresholding			BackDisc			
corpus name	cols	min words	rows	Sp	G	Err	T	Err	T	Sp	Err	W T	H T	Sp	Err
	(#)	(#)	(#)	(%)		(%)	(s)	(%)	(s)	(%)	(%)	(s)	(s)	(%)	(%)
abc	100	0	3189	13	10	72	2	85	0	1	97	0	0	2	92
abc	100	0	3189	13	50	30	10	68	0	5	77	1	0	7	64
abc	100	100	2627	13	10	72	2	85	0	2	97	0	0	2	94
abc	100	100	2627	13	50	30	9	68	0	6	78	1	0	7	66
abc	300	0	3189	8	10	82	4	89	0	0	99	0	0	0	97
abc	300	0	3189	8	50	60	17	81	0	1	91	1	0	2	86
abc	300	100	2627	9	10	83	4	89	0	0	99	0	0	0	97
abc	300	100	2627	9	50	60	14	82	0	1	92	1	0	2	86
brown	100	0	15667	4	10	75	9	89	0	0	93	3	0	0	90
brown	100	0	15667	4	50	33	49	71	0	2	75	5	0	2	62
brown	100	100	3530	9	10	78	2	88	0	1	95	0	0	1	92
brown	100	100	3530	9	50	34	12	71	0	4	75	1	0	5	63
brown	300	0	15667	3	10	86	18	93	0	0	97	4	0	0	95
brown	300	0	15667	3	50	63	72	85	0	0	88	7	0	0	82
brown	300	100	3530	6	10	87	5	93	0	0	98	0	0	0	96
brown	300	100	3530	6	50	65	17	84	0	1	89	1	0	1	84
gutenberg	100	0	47887	4	10	71	29	88	0	0	93	16	0	0	89
gutenberg	100	0	47887	4	50	30	151	70	1	2	74	24	0	3	61
gutenberg	100	100	5523	11	10	75	3	86	0	1	97	1	0	1	93
gutenberg	100	100	5523	11	50	31	18	69	0	5	76	3	0	7	62
gutenberg	300	0	47887	2	10	82	61	92	1	0	96	23	0	0	94
gutenberg	300	0	47887	2	50	58	210	82	1	0	86	35	0	0	80
gutenberg	300	100	5523	7	10	84	8	91	0	0	99	1	0	0	97
gutenberg	300	100	5523	7	50	61	28	82	0	1	89	3	0	2	82
inaugural	100	0	1515	8	10	71	1	86	0	1	95	0	0	1	91
inaugural	100	0	1515	8	50	28	5	68	0	4	74	0	0	5	62
inaugural	100	100	509	12	10	73	0	85	0	1	98	0	0	1	95
inaugural	100	100	509	12	50	26	2	67	0	5	78	0	0	7	68
inaugural	300	0	1515	5	10	82	2	90	0	0	98	0	0	0	96
inaugural	300	0	1515	5	50	55	8	81	0	1	88	0	0	1	83
inaugural	300	100	509	8	10	82	1	90	0	0	99	0	0	0	98
inaugural	300	100	509	8	50	52	3	80	0	2	93	0	0	2	90
movie_reviews	100	0	2000	16	10	80	1	86	0	0	99	0	0	1	96
movie_reviews	100	0	2000	16	50	36	7	71	0	6	79	1	0	8	69
movie_reviews	100	100	1999	16	10	80	1	86	0	1	99	0	0	1	96
movie_reviews	100	100	1999	16	50	35	7	71	0	6	80	1	0	8	69
movie_reviews	300	0	2000	13	10	89	2	89	0	0	99	0	0	0	99
movie_reviews	300	0	2000	13	50	67	11	83	0	2	94	1	0	2	90
movie_reviews	300	100	1999	13	10	89	3	89	0	0	100	0	0	0	100
movie_reviews	300	100	1999	13	50	67	11	83	0	1	95	1	0	2	90
reuters	100	0	11887	10	10	67	7	84	0	1	97	3	0	2	94
reuters	100	0	11887	10	50	28	40	68	0	4	77	6	0	6	63
reuters	100	100	5903	13	10	68	4	84	0	2	97	1	0	3	93
reuters	100	100	5903	13	50	28	19	67	0	6	77	3	0	8	64
reuters	300	0	11887	6	10	77	18	89	0	0	98	4	0	0	96
reuters	300	0	11887	6	50	55	60	80	0	1	90	7	0	2	85
reuters	300	100	5903	9	10	79	8	89	0	0	99	2	0	0	96
reuters	300	100	5903	9	50	57	32	80	0	2	92	3	0	2	87
webtext	100	0	3247	6	10	68	2	85	0	1	93	0	0	1	89
webtext	100	0	3247	6	50	23	10	65	0	3	70	1	0	4	56
webtext	100	100	627	9	10	59	0	80	0	2	93	0	0	3	90
webtext	100	100	627	9	50	14	4	57	0	6	67	0	0	7	52
webtext	300	0	3247	3	10	77	4	89	0	0	96	0	0	0	94
webtext	300	0	3247	3	50	49	16	78	0	1	84	1	0	1	77
webtext	300	100	627	6	10	70	1	85	0	1	95	0	0	1	92
webtext	300	100	627	6	50	41	4	73	0	2	86	0	0	3	79