Generalized Matrix Factorizations as a Unifying Framework for Pattern Set Mining: Complexity Beyond Blocks

Pauli Miettinen

Max-Planck-Institut für Informatik
Saarbrücken, Germany
pauli.miettinen@mpi-inf.mpg.de

Abstract. Matrix factorizations are a popular tool to mine regularities from data. There are many ways to interpret the factorizations, but one particularly suited for data mining utilizes the fact that a matrix product can be interpreted as a sum of rank-1 matrices. Then the factorization of a matrix becomes the task of finding a small number of rank-1 matrices, sum of which is a good representation of the original matrix. Seen this way, it becomes obvious that many problems in data mining can be expressed as matrix factorizations with correct definitions of what a rank-1 matrix and a sum of rank-1 matrices mean. This paper develops a unified theory, based on generalized outer product operators, that encompasses many pattern set mining tasks. The focus is on the computational aspects of the theory and studying the computational complexity and approximability of many problems related to generalized matrix factorizations. The results immediately apply to a large number of data mining problems, and hopefully allow generalizing future results and algorithms, as well.

1 Introduction

One of the most fundamental tasks in data mining is to explain (or summarize) a data set using a collection of simple and easy-to-understand structures (commonly referred to as *regularities* or *patterns*). A *block* of some kind has been the predominant type of patterns sought by many data mining algorithms; a clique (or quasi-clique) in a network or a frequent itemset or a tile in transaction data are all blocks, or more precisely, (binary) rank-1 (sub-)matrices. Consequently, a collection of these blocks summarizing the data can be seen as a matrix factorization of the data matrix [27].

But in recent years, increased emphasis has been put to patterns that go 'beyond blocks', for example, nested submatrices [17], taxonomies [18], stars, biclique cores, or chains [19], or hyperbolic subgraphs [3] (see Figure 1 for examples of some of these patterns). At first, it might seem like these patterns share very little, if anything, with simple blocks; they are, for example, not rank-1. It seems, then, that we have to re-do much of the work we have already done in analysing the computational aspects of mining collections of blocks.

More in-depth study, however, starts revealing various similarities between these patterns that are 'beyond blocks' and the simple blocks. The intuition of expressing a data sets using a collection of simple patterns, for example, is the same. Consequently, we can *still* consider the patterns as 'rank-1' sub-matrices, and the whole summarization as a form of matrix factorization – we only need new definitions of ranks and matrix factorizations.

In this paper I propose *generalized outer products* as a unifying framework to express different kinds of patterns. As the name implies, it generalizes the outer product of two vectors (i.e. the operator used to express cliques and other block patterns). The new generalized definitions of matrix rank and factorization follow from the outer product, as we will see in Section 3.

The framework works over any semi-ring, but for the sake of concreteness, the examples are presented using a fixed set of binary patters, introduced in Example 1 and Figure 1. After we have seen the definition of the framework (Section 3), we study how some common concepts, such as the matrix rank, behave under it (Section 3.2).

The framework alone is not useful for data mining researchers, though. To that end, it must help the researchers to obtain interesting results. To demonstrate the proposed framework's capability to do that, we see a series of general results regarding the computational complexity (Section 4) and approximability (Section 5) of some fundamental problems related to the generalized decompositions in binary matrices. These results will immediately yield corresponding results for any pattern fitting to the framework.

The purpose of this paper is to develop the framework and to demonstrate its usefulness via number of general results. As a consequence, some of the results we will see are already known in the literature in some specific cases; indeed, some of the results have been presented for multiple special cases – an unnecessary repetition that can be avoided with the proposed framework. The goal of this paper is *not* to present novel algorithms for mining the patterns. While it is expected that the framework facilitates developing of general algorithms, more in-depth studies to that end are left for future work. That said, we do see number of existing algorithms (mostly in Section 5) that can be used to solve certain general problems in the proposed framework with provable performance guarantees.

Before moving on, let us briefly discuss some related work.

2 Related work

Finding patterns from data is in the core of data mining, with frequent itemset mining being an early and prominent example. Much of the research focus has nowadays sifted from finding all the patterns to finding an interesting subset of them, with the interestingness being defined either in combinatorial [15, 34], information-theoretic [10, 33], or other means. Similar problems were also studied, for example, in formal concept analysis [6] and role mining [12].

What all of the aforementioned work shares is that they aim at describing a binary matrix using a set of rank-1 binary sub-matrices. The requirement for sub-

matrices is strict: all these methods are restricted to rank-1 matrices that appear in the input data as such (known as from-below or dominated decompositions). When this requirement is lifted, the problem is usually referred to as Boolean matrix factorization [22, 25], although especially earlier other names were also used [28].

A seemingly unrelated line of research grew from the problem of finding communities from graphs. Traditionally, communities were considered exclusively as cliques (or bicliques, in case of bipartite graphs), corresponding again to rank-1 sub-matrices of the adjacency matrix. While this connection is often not made explicitly, it can be – and has been [35] – used to design community-detection algorithms, especially for the overlapping case.

Mere cliques (or bicliques), however, might not be enough to properly explain the interesting communities in the graphs [19], and recently many real-world communities were found out to be more hyperbolic than clique-like [3]. The requirement for communities more complex than simply cliques has been encapsulated to the "beyond blocks" slogan.

Communities or itemsets or rank-1 matrices are not the only kind of patterns data miners are interested about, of course. Patterns such as nested or banded submatrices [17] or taxonomies [18], among many others, are equally well expressed in the generalized framework of this paper.

3 Definitions

Throughout this paper, upper-case bold symbols ($\boldsymbol{A}$) will be used to denote matrices, lower-case bold symbols ($\boldsymbol{a}$) denote vectors, and lower-case normal symbols (a) denote scalars. If n is an integer, the shorthand notation $[n]$ is used for set $\{1, 2, \ldots, n\}$.

For any matrix $\boldsymbol{A}$ (binary or not), $|\boldsymbol{A}|$ denotes the number of non-zero elements in it.

We work over algebraic structure $\mathbb{T} = (T, \boxplus, \boxtimes, 0, 1)$. The binary operator $\boxplus$ is called addition and the binary operator $\boxtimes$ is called multiplication, with $0 \in T$ and $1 \in T$ being their respective identity elements. $\mathbb{T}$ is required to be at least a semiring, that is, $\boxplus$ is commutative and $\boxtimes$ distributes over $\boxplus$.

Before going forward to the definitions, let us see different types of patterns that will be used in examples throughout the paper.

Example 1. A *biclique*, a *binary rank-1 matrix*, and a *(combinatorial) tile* all refer to the same kind of pattern: a submatrix full of 1s, that is, a block. A *star* (Figure 1, left) and a *biclique core* (Figure 1, middle-left) are forms of patterns in (undirected) graphs: star represents a collection of vertices that are connected to each other only via a single hub vertex, while a biclique core represents a set of vertices that form a complete bipartite graph. A *chain* (Figure 1, middle-right) is a set of vertices where each vertex is connected only to the next one, while *nested matrix* (Figure 1, right) is a bipartite graph where each subsequent vertice's neighbors are a subset of the previous one's neighbors.

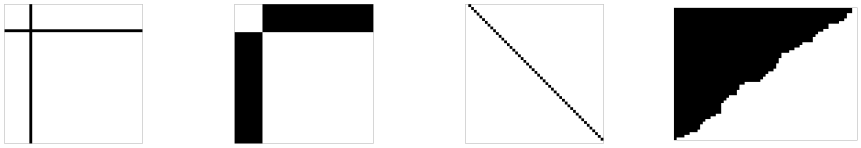

Fig. 1. Different types of patters, from left to right: star, biclique core, chain, and nested. The first three are symmetric square matrices; nested is asymmetric general matrix.

Note that in Figure 1, all matrices are permuted for maximum readability; in general, no particular ordering of the rows or columns is required. ◊

3.1 Generalized Outer Product

The definition of the generalized outer product is the core of the proposed framework. Similar definitions have appeared earlier, but to the best of the author's knowledge, this exact definition of generalized outer product has not been proposed earlier.

Definition 1. *The* generalized outer product operator *of two vectors* $\boldsymbol{x} \in \mathbb{T}^n$ *and* $\boldsymbol{y} \in \mathbb{T}^m$ *with parameters* $\theta \in \Theta$ *is a function* $o : \mathbb{T}^n \times \mathbb{T}^m \times \Theta \to \mathbb{T}^{n \times m}$ *such that for all* $(i, j) \in [n] \times [m]$*, if* $x_i = 0$ *or* $y_j = 0$*, then* $o(\boldsymbol{x}, \boldsymbol{y}, \theta)_{ij} = 0$*.*

It is helpful to consider the outer products of the four patterns in Figure 1:

Example 2. The star pattern can be generated using generalized outer product $o_s(\boldsymbol{x}, \boldsymbol{x}, k) = (a_{ij})$, where

$$a_{ij} = \begin{cases} 1 & \text{if } x_i = 1 \text{ and } x_j = 1 \text{ and either } i = k \text{ or } j = k \\ 0 & \text{otherwise .} \end{cases}$$

This can be interpreted as follows: binary vector $\boldsymbol{x}$ selects the rows and columns (e.g. vertices) that participate in the pattern, while parameter k chooses the vertex that is the centre of the star. Naturally, if $x_k = 0$ this yields empty pattern. We could require that $x_k = 1$ without significant changes to anything that follows. Similar requirements could be applied to many of the following patterns, as well, but they are not stated for the sake of brevity.

The biclique core pattern can be generated using generalized outer product $o_{bc}(\boldsymbol{x}, \boldsymbol{x}, I \subset [n]) = (a_{ij})$, where

$$a_{ij} = \begin{cases} 1 & \text{if } x_i = 1 \text{ and } x_j = 1 \text{ and exactly one of } i \in I \text{ or } j \in I \text{ holds} \\ 0 & \text{otherwise .} \end{cases}$$

The outer product generating the chain pattern is $o_c(\boldsymbol{x}, \boldsymbol{y}) = (a_{ij})$, where $a_{ij} = 1$ if $x_i = 1$, $y_j = 1$, and $j = i + 1$, and $a_{ij} = 0$ otherwise.

The outer product for nested pattern is $o_n(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}) = (a_{ij})$, where $\boldsymbol{s} \in [m]^n$ defines a *step function* and

$$a_{ij} = \begin{cases} 1 & \text{if } x_i = 1 \text{ and } y_j = 1 \text{ and } j \leq s_i \\ 0 & \text{otherwise .} \end{cases}$$

Notice that for the resulting pattern to be valid, $s_i \leq s_j$ for all $i > j$ if we are looking for *direct* nested submatrices; if we want to find general nested submatrices, the indices must be permuted appropriately (see [17] for more information on direct and general nested patterns). ◊

In the above examples, all outer products were defined element-wise. The generalized outer products that can be defined element-wise form an important sub-class of generalized outer products, called *decomposable* outer products:

Definition 2. *A generalized outer product operator o is* decomposable *if we have that $o(\boldsymbol{x}, \boldsymbol{y}, \theta)_{ij} = f(x_i, y_j, i, j, \theta)$ for all i and j. We say that o* is decomposable to f.

The indices i and j can be considered as two additional parameters in θ and will often be omitted for brevity's sake.

Another common feature shared with most of the above examples is that the outer product is with vector $\boldsymbol{x}$ itself. This ensures that the product is *symmetric*.

Definition 3. *An outer product $o(\boldsymbol{x}, \boldsymbol{y}, \theta)$ is* symmetric *if $\boldsymbol{x} = \boldsymbol{y}$ and $o(\boldsymbol{x}, \boldsymbol{y}, \theta) = o(\boldsymbol{x}, \boldsymbol{y}, \theta)^T$.*

For decomposable outer products, it is enough to require that $\boldsymbol{x} = \boldsymbol{y}$ as the other constraint follows automatically.

3.2 Generalized Rank

A common way to measure the complexity of the structure of a matrix is its *rank*. Multiple equivalent definitions of a matrix rank exist under normal linear algebra, but most of them do not generalize well to our case. We shall now generalize the so-called *Schein rank* and study some properties of the resulting general rank.

Definition 4. *Matrix $\boldsymbol{M} \in \mathbb{T}^{n \times m}$ has outer product operator o induced* rank, $\text{rank}_o(\boldsymbol{M}) = 1$ *if there exists vectors $\boldsymbol{x} \in \mathbb{T}^n$ and $\boldsymbol{y} \in \mathbb{T}^m$ and parameters $\theta \in \Theta$ such that $\boldsymbol{M} = o(\boldsymbol{x}, \boldsymbol{y}, \theta)$.*

For brevity, the outer product will be omitted from the rank when it is clear from the context. Hence, if matrix $\boldsymbol{M}$ is rank-1, it implicitly means that its o-induced rank is 1.

Definition 5. *The (generalized)* sum *of two matrices $\boldsymbol{A} \in \mathbb{T}^{n \times m}$ and $\boldsymbol{B} \in \mathbb{T}^{n \times m}$ is the element-wise sum $\boldsymbol{A} \boxplus \boldsymbol{B}$ using the summation $\boxplus$ of $\mathbb{T}$. That is, $(\boldsymbol{A} \boxplus \boldsymbol{B})_{ij} = a_{ij} \boxplus b_{ij}$.*

Definition 6. *A set* $D = \{\boldsymbol{F}_i \in \mathbb{T}^{n\times m} : \mathrm{rank}_o(\boldsymbol{F}_i) = 1\}$ *is the* decomposition *(under* o *and* $\boxplus$*) of a matrix* $\boldsymbol{M} \in \mathbb{T}^{n\times m}$ *if*

$$\boldsymbol{M} = \boldsymbol{F}_1 \boxplus \boldsymbol{F}_2 \boxplus \cdots \boxplus \boldsymbol{F}_{|D|} \ . \tag{1}$$

The size *of the decomposition* D *is* $|D|$*.*

If D is a decomposition of $\boldsymbol{A}$ and o is decomposable to f, we get the familiar element-wise matrix product form

$$a_{ij} = \boxplus_{k=1}^{|D|} f(x_{ik}, y_{jk}, \theta) \ , \tag{2}$$

where $o(\boldsymbol{x}_k, \boldsymbol{y}_k, \theta) = \boldsymbol{F}_k \in D$ for all $i \in [|D|]$.

Hence, we can define the matrix product for decomposable outer products.

Definition 7. *If* o *is decomposable to* f*, the* matrix product *of* $\boldsymbol{A} \in \mathbb{T}^{n\times k}$ *and* $\boldsymbol{B} \in \mathbb{T}^{k\times m}$ *(under* o*,* θ*, and* $\boxplus$*) is defined element-wise as*

$$(\boldsymbol{A} \boxtimes_\theta \boldsymbol{B})_{ij} = \boxplus_{l=1}^{k} f(a_{il}, b_{lj}, \theta) \ . \tag{3}$$

Definition 8. *Let* o *be decomposable to* f*. The operator* $\langle\cdot,\cdot\rangle : \mathbb{T}^n \times \mathbb{T}^n \to \mathbb{T}$ *is* inner product *if it satisfies the following rules for all* $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{T}^n$ *and* $\alpha \in \mathbb{T}$*:*

$$\langle \boldsymbol{x}, \boldsymbol{y}\rangle = \langle \boldsymbol{y}, \boldsymbol{x}\rangle \tag{4a}$$
$$\langle \alpha \boxtimes \boldsymbol{x}, \boldsymbol{y}\rangle = \alpha \boxtimes \langle \boldsymbol{x}, \boldsymbol{y}\rangle \tag{4b}$$
$$\langle \boldsymbol{x} \boxplus \boldsymbol{y}, \boldsymbol{z}\rangle = \langle \boldsymbol{x}, \boldsymbol{z}\rangle \boxplus \langle \boldsymbol{y}, \boldsymbol{z}\rangle \tag{4c}$$
$$\langle \boldsymbol{x}, \boldsymbol{x}\rangle \geq 0 \tag{4d}$$
$$\langle \boldsymbol{x}, \boldsymbol{x}\rangle = 0 \Rightarrow \boldsymbol{x} = 0 \ . \tag{4e}$$

If decomposable o *induces an inner product, we say* o *is* inner-product compatible*.*

Notice that the decomposability is the crucial element here.

Definition 9. *The* rank *(over* o*) of* $\boldsymbol{A} \in \mathbb{T}^{n\times m}$*,* $\mathrm{rank}_o(\mathbb{T})$*, is the smallest integer* k *such that there exists a decomposition* D *of* $\boldsymbol{A}$ *with size* k*. If no decomposition exist,* $\mathrm{rank}_o(\mathbb{T}) = \infty$*.*

A *singleton* matrix has exactly one non-zero value, and is characterized by triple (i, j, α), corresponding to a matrix $\boldsymbol{A}$ for which $a_{pq} = \alpha$ if $i = p$ and $j = q$ and $a_{pq} = 0$ otherwise. If an outer product operator o can generate all n-by-m singleton matrices $(i, j, \alpha) \in [n] \times [m] \times \mathbb{T}$ (it is *singleton-generating*), the rank of any $\boldsymbol{A} \in \mathbb{T}^{n\times m}$ induced by o is bounded by $\mathrm{rank}_o(\boldsymbol{A}) \leq |\boldsymbol{A}| \leq nm$.

Example 3. The star outer product o_s is singleton-generating: we can set $\boldsymbol{x}$ to have exactly one non-zero and set k to the index of that non-zero. These singleton-stars can be used to represent any symmetric binary matrix $\boldsymbol{A}$. The biclique core outer product o_{bc} is not singleton-generating, however, and it can induce infinite ranks. Consider, for example, the 2-by-2 identity matrix $\boldsymbol{I}_2$:

$$\boldsymbol{I}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} .$$

As every vertex is connected to itself, and only to itself, there are no bipartite graphs and $\boldsymbol{I}_2$ cannot be expressed with a set of biclique cores, that is, $\operatorname{rank}_{o_{bc}}(\boldsymbol{I}_2) = \infty$. ◊

4 Computational Complexity

We will now move to the applications of the proposed framework. Specifically, we will consider some results regarding the computational complexity and approximability of problems related to generalized decompositions in binary matrices.

4.1 Rank-1 Submatrices

Let us start by studying problems regarding finding the largest rank-1 matrix. It is a very common sub-problem in almost all algorithms that need to find a set of patterns. For the results, we need the definition of *hereditary* outer product.

Definition 10. *Binary outer product operator o is* hereditary *if the class of rank-1 matrices induced by it is closed under permutation and deletion of rows and columns.*

Example 4. Stars, biclique cores, and nested matrices are closed under permutations and deletions of rows and columns, and consequently their outer products are hereditary. Chains are not closed under deletion of rows and columns, and the outer product is not hereditary.

The exact definition of the problems depends on how we define "large". We start by studying the case where large pattern is one with large circumference.

Problem 1. In the *binary maximum-circumference o-induced rank-1 submatrix problem* we are given a binary matrix $\boldsymbol{A} \in \{0,1\}^{n \times m}$ and our task is to find vectors $\boldsymbol{x} \in \{0,1\}^n$ and $\boldsymbol{y} \in \{0,1\}^m$ and parameters θ such that $o(\boldsymbol{x}, \boldsymbol{y}, \theta)$ is dominated by $\boldsymbol{A}$ and we maximize the (half-) *circumference* $|\boldsymbol{x}| + |\boldsymbol{y}|$.

Proposition 1. *Let $\mathcal{M}_o$ be the family of all o-induced rank-1 matrices (of any size), where o is hereditary. If the number of distinct rows (or columns) in matrices of $\mathcal{M}_o$ is unbounded, the binary maximum-circumference o-induced rank-1 submatrix problem is* NP*-hard; if the number of distinct rows (or columns) is bounded, the problem can be solved in polynomial time.*

The result follows straight forwardly from Corollary 4 of Yannakakis [36]. Many interesting patterns have bounded number of distinct rows, for example, bicliques have exactly two kinds of rows: the rows corresponding to the nodes that are in the biclique, and the rows corresponding to the nodes that are not in it. On the other hand, for example nested matrices have unbounded number of distinct rows. Notice also that one cannot have bounded number of distinct rows but unbounded number of distinct columns (or vice versa) in a binary matrix: for k distinct rows one cannot have more than 2^k distinct columns.

The *symmetric* maximum-circumference rank-1 submatrix problem is like its asymmetric case, but we require the input matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$ and the outer product o be symmetric (and hence the outer product is of type $o(\boldsymbol{x}, \boldsymbol{x}, \theta)$). Adding the symmetry requirement makes the problem harder, as now it is enough that there are infinitely-many rank-1 submatrices and infinitely many matrices with higher ranks.

Proposition 2. *Let $\mathcal{S}_o$ be the family of all binary matrices generated by hereditary binary symmetric outer product $o(\boldsymbol{x}, \boldsymbol{x}, \theta)$ and let $\mathcal{S}_o^c$ be the family of all symmetric binary matrices not in $\mathcal{S}_o$. If $|\mathcal{S}_o| = |\mathcal{S}_o^c| = \infty$, then finding the maximum-circumference o-induced symmetric rank-1 submatrix is* NP*-hard.*

The result follows from Lewis and Yannakakis [21].

Another possible definition for "large" is to study the area: the maximum-area rank-1 problems ask to maximize the area instead of the circumference.

Problem 2. In the *binary maximum-area o-induced rank-1 submatrix problem* we are given a binary matrix $\boldsymbol{A} \in \{0,1\}^{n \times m}$ and our task is to find vectors $\boldsymbol{x} \in \{0,1\}^n$ and $\boldsymbol{y} \in \{0,1\}^m$ and parameters θ such that $o(\boldsymbol{x}, \boldsymbol{y}, \theta)$ is dominated by $\boldsymbol{A}$ and we maximize the *area* $|\boldsymbol{x}|\,|\boldsymbol{y}|$.

The symmetric binary maximum-area rank-1 submatrix problem is defined analogously. As the area of a symmetric rank-1 matrix is simply $|\boldsymbol{x}|^2$, the hardness of the symmetric problems is a straight forward corollary of Proposition 2.

Proposition 3. *The binary symmetric maximum-area rank-1 submatrix problem is* NP*-hard exactly when the binary symmetric maximum-circumference rank-1 problem is.*

Proof. The maximum-circumference problem asks us to decide if the given matrix has a symmetric rank-1 submatrix such that $2\,|\boldsymbol{x}| \geq t$, while in the maximum-area problem the condition is that $|\boldsymbol{x}|^2 \geq t'$. Thus, we can solve the maximum-circumference problem by solving the maximum-area problem with $t' = t^2/4$. □

For the asymmetric case this trivial correspondence does not necessarily hold, as can be readily witnessed by noticing that the binary maximum-area rank-1 submatrix problem under standard algebra corresponds to the maximum-edge biclique, and hence is NP-hard [29].

The third variant of binary maximum rank-1 submatrices are the *maximum-content* submatrices, that is, the submatrices with maximum number of non-zeros (in symmetric matrices, this corresponds to maximum-edge subgraphs). The complexity of these problems can often be reduced to the complexity of the maximum-area or maximum-circumference problems.

Proposition 4. *If there exists a set of parameters θ such that the number of non-zeros in $o(\boldsymbol{x}, \boldsymbol{y}, \theta)$ depends only on circumference $|\boldsymbol{x}|+|\boldsymbol{y}|$ (resp. area $|\boldsymbol{x}|\,|\boldsymbol{y}|$), then finding the maximum-content rank-1 submatrix is* NP*-hard if finding the maximum-circumference (resp. maximum-area) rank-1 submatrix is* NP*-hard.*

4.2 Selecting Some Rank-1 Submatrices

We now turn to problems where we are given a set of rank-1 matrices, and our task is to select some of them (e.g. to be presented to the user). This is also a common subproblem in many pattern set mining algorithms, where first a set of candidate solutions is generated, and then a final selection is performed from that set.

Problem 3. In the *smallest binary sub-decomposition problem* we are given a matrix $\boldsymbol{A} \in \{0,1\}^{n \times m}$ and its decomposition $D = \{\boldsymbol{F}_i \in \{0,1\}^{n \times m} : \operatorname{rank}_o(\boldsymbol{F}_i) = 1\}$ and our task is to find the smallest subset $C \subseteq D$ that is still a valid decomposition, i.e., $\boxplus_{\boldsymbol{F} \in C} \boldsymbol{F} = \boldsymbol{A}$.

Proposition 5. *If $\boxplus$ is either logical OR, logical AND, or logical XOR, the smallest binary sub-decomposition problem is* NP*-hard.*

Proof. We study the cases separately.

OR: This case is very similar to the Tiling databases [15], and we present the reduction only for the sake of completeness. The reduction is from the *minimum set cover* problem [14]. Let $(U, \mathcal{S} \subset 2^U)$ be a set system. Let $\boldsymbol{a} \in \{0,1\}^n$ be an all-1s vector where $|U| = n$, and for each $S \in \mathcal{S}$, define vector $\boldsymbol{f}^{(S)} \in D$ to be the *characteristic vector* of S, that is $f_i^{(S)} = 1$ if $u_i \in S$ and $f_i^{(S)} = 0$ otherwise. It is trivial to see that the smallest sub-decomposition $C \subseteq D$ for which $\bigvee_{\boldsymbol{f}^{(S)} \in C} \boldsymbol{f}^{(S)}$ is equivalent to the minimum set cover.

AND: This is similar to above, except that the reduction constructs the complements of $\boldsymbol{a}$ (which is all-0s) and $\boldsymbol{f}^{(S)}$s and the proof follows from De Morgan's laws.

XOR: In the *decoding of linear codes* problem [14], we are given a binary matrix $\boldsymbol{B} \in \{0,1\}^{n \times k}$ and a binary vector $\boldsymbol{a} \in \{0,1\}^k$ and we need to find a binary vector $\boldsymbol{x} \in \{0,1\}^k$ minimizing $|\boldsymbol{x}|$ such that $\bigoplus_{j=1}^{k} x_j b_{ij} = a_i$ for all $i \in [n]$, where $\oplus$ is the logical XOR operatior. To reduce this to the sub-decomposition problem, it is enough to notice that if we take the column vectors $\boldsymbol{b}_j$ of matrix $\boldsymbol{B}$ as the factors in D we have the smallest binary sub-decomposition problem.[1] □

[1] A minor technicality is that $\boldsymbol{B}$ for which $\bigoplus_j \boldsymbol{b}_j \neq \boldsymbol{a}$ does not yield to a valid input to the smallest sub-decomposition problem. This can be solved by adding a new column $\boldsymbol{c} = \bigoplus_j \boldsymbol{b}_j$ to $\boldsymbol{B}$, and by adding one row to $\boldsymbol{B}$ and $\boldsymbol{a}$; this row has value 1 in $\boldsymbol{a}$ and $\boldsymbol{c}$, and is 0 in other columns of $\boldsymbol{B}$.

These problems can also be seen as generalizations of a test for linear independency: in some sense, what we need to remove are the factors that are not independent from the others.

4.3 Minimum-Error Sub-Decompositions

In many applications we can assume the input data contains noise, and has high (or even infinite) o-induced rank. In these situations, we might be more interested on approximate decompositions, and instead of finding the smallest exact sub-decomposition, we want to find a sub-decomposition that induces the minimum error.

Problem 4. In the *minimum-error binary sub-decomposition problem* we are given a matrix $\boldsymbol{A} \in \{0,1\}^{n \times m}$ and a set $D = \{\boldsymbol{F}_i \in \{0,1\}^{n \times m} : \operatorname{rank}_o(\boldsymbol{F}_i) = 1\}$ and our task is to find subset $C \subseteq D$ of size k that minimizes

$$\left| \boxplus_{\boldsymbol{F} \in C} \boldsymbol{F} - \boldsymbol{A} \right| . \tag{5}$$

Using the Hamming distance, as in (5), is natural in case of binary decompositions; other error measures are of course possible, especially for matrices taking non-binary values. These problems are no easier than the smallest sub-decomposition problems.

Proposition 6. *If $\boxplus$ is either logical OR, logical AND, or logical XOR, the minimum-error binary decomposition problem is* NP*-hard.*

Proof. We again work case-by-case:

OR: In the *basis usage* problem we are given a binary vector $\boldsymbol{a} \in \{0,1\}^n$ and a binary matrix $\boldsymbol{B} \in \{0,1\}^{n \times k}$, and our task is to find a binary vector $\boldsymbol{x} \in \{0,1\}^k$ such that $\left| \boldsymbol{a} - \bigvee_{i=1}^k x_i \boldsymbol{b}_i \right|$ is minimized, where $x_i \boldsymbol{b}_i$ yields all-0s vector if $x_i = 0$, and $\boldsymbol{b}$ if $x_i = 1$. This clearly a special case of minimum-error binary sub-decomposition, and the claim follows as basis usage problem is NP-hard [25].

AND: This case again follows from De Morgan's laws by taking complements and using the above reduction.

XOR: In the *nearest codeword* problem we are given a binary vector $\boldsymbol{a} \in \{0,1\}^n$ and a binary matrix $\boldsymbol{B} \in \{0,1\}^{n \times k}$, and our task is to find a binary vector $\boldsymbol{x} \in \{0,1\}^k$ such that $\left| \boldsymbol{a} - \oplus_{i=1}^k x_i \boldsymbol{b}_i \right|$ is minimized. This, again, is a special case of minimum-error binary sub-decomposition, and the claim follows as the nearest codeword problem is NP-hard [4]. □

4.4 Deciding the Rank

Deciding the (generalized) rank of a matrix is a fundamental question. Unfortunately, the complexity of deciding the rank depends on the interplay between the underlying algebraic structure $\mathbb{T}$ and the generalized outer product o, as the following examples illustrate.

Example 5. Consider binary n-by-m matrices and the normal outer product $o(\boldsymbol{x}, \boldsymbol{y}, \theta) = \boldsymbol{x}\boldsymbol{y}^T$. If the summation $\boxplus$ is the logical OR operator, the rank is the *Boolean rank* of the matrix, and consequently NP-hard (see, e.g. [28]). If, however, the summation $\boxplus$ is the logical XOR operator, finding the rank can be done in polynomial time [31]. $\Diamond$

We say binary outer product $o(\boldsymbol{x}, \boldsymbol{y}, \theta)$ *subsumes* bicliques if there exists parameters θ such that $o(\boldsymbol{x}, \boldsymbol{y}, \theta) = \boldsymbol{x}\boldsymbol{y}^T$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$.

Corollary 1. *Let $\mathbb{T} = (\{0,1\}, \vee, \boxtimes, 0, 1)$. Finding the o-induced rank of $\boldsymbol{A} \in \mathbb{T}^{n \times m}$ is* NP*-hard if o subsumes the bicliques.*

4.5 Minimum-Error Approximate Decompositions

The *minimum-error fixed-rank decompositions* are to the rank-decision problems what the minimum-error fixed-size sub-decompositions are to the smallest sub-decompositions.

Problem 5. In the *minimum-error fixed-rank binary decomposition* we are given a matrix $\boldsymbol{A} \in \{0,1\}^{n \times m}$ and an integer k, and our task is to find set $D = \{\boldsymbol{F}_i \in \{0,1\}^{n \times m} : \mathrm{rank}_o(\boldsymbol{F}_i) = 1, i \in [k]\}$ that minimizes

$$\left| \bigboxplus_{\boldsymbol{F} \in D} \boldsymbol{F} - \boldsymbol{A} \right| . \tag{6}$$

In the decision version of Problem 5, the input is prepended with parameter $t \in \mathbb{R}$ and instead of minimizing (6), the task is to decide if there exists D such that

$$\left| \bigboxplus_{\boldsymbol{F} \in D} \boldsymbol{F} - \boldsymbol{A} \right| \leq t . \tag{7}$$

It is easy to see that the complexity of these problems is no easier than that of the related rank-decision problems:

Proposition 7. *If computing the o-rank is* NP*-hard, so is computing the minimum-error approximate decomposition.*

Proof. If we set $t = 0$ in (7), the o-rank of $\boldsymbol{A}$ is the least k for which we have positive answer. $\square$

5 Approximability

As we saw, most problems related to binary generalized outer products are NP-hard. Approximation algorithms are a common recourse to the NP-completeness, and in this section we will study the approximability of some of the problems studied above. As we shall see, many – but not all – problems are hard even to approximate, giving a post hoc justification to the various heuristics employed in the prior work.

5.1 Approximating Smallest Sub-Decompositions

We start with the problem that is easiest to approximate:

Proposition 8. *If $\boxplus = \vee$, the smallest binary sub-decomposition can be approximated to within $\ln n$ (and no better) in polynomial time.*

Proof. We reduce the problem to the minimum set cover in an approximation-preserving way. Let $U = \{(i,j) : a_{ij} = 1\}$ be the set of all locations of $\boldsymbol{A}$ that are 1. For every matrix $\boldsymbol{F}_k \in D$, define set $F_k = \{(i,j) : (\boldsymbol{F}_k)_{ij} = 1\}$ as the locations of ones in $\boldsymbol{F}_k$ and let the collection $\mathcal{D} = \{F_k : k = 1, \ldots, |D|\}$. As D is a decomposition of $\boldsymbol{A}$, it is guaranteed that $\bigcup_{k=1}^{|D|} F_k = U$. The task of finding the smallest sub-decomposition of D is equivalent to finding the smallest sub-collection $\mathcal{C} \subseteq \mathcal{D}$ such that $\bigcup_{F_k \in \mathcal{C}} F_k = U$, that is, the minimum set cover problem. As the reduction preserves the value of the optimization target, it preserves the approximability and hence we can use, for example, the famous greedy $\Theta(\ln n)$-approximation algorithm [16]. On the other hand, the reduction in the proof of Proposition 5 is also approximation-preserving, and consequently, the $\ln n$ bound is tight unless P = NP [13]. □

The approximation-preserving reduction to and from set cover, together with the result of Simon [32], gives us the following interesting corollary:

Corollary 2. *Given an algorithm that, in every iteration of the greedy algorithm, selects the factor $\boldsymbol{F}_i \in D$ that* approximates *the best choice by a factor of $O(h(n))$, we can approximate the smallest binary sub-decomposition by a factor of $O(h(n) \ln n)$.*

This corollary can be very useful when the decomposition D is given only implicitly, and cannot be exhaustively searched for the best solution in every iteration. For example, the tiling algorithm of [15] needs – in principle – to search every closed itemset of the input data in every iteration. To avoid that, the authors resolve to clever heuristics, but with the cost of approximation guarantees. If the heuristic algorithm could be replaced with one with provable approximation guarantees, Corollary 2 would give us an overall approximation guarantee.

When $\boxplus = \oplus$, the problem becomes harder to approximate. The *minimum weight codeword* problem is similar to the aforementioned decoding of linear codes problem. In the former, we are given a matrix $\boldsymbol{B} \in \{0,1\}^{n \times k}$ and an all-zeros vector $\boldsymbol{a} \in \{0,1\}^n$, and our task is to find a *non-empty* vector $\boldsymbol{x} \in \{0,1\}^n$ such that $\boldsymbol{Bx} = \boldsymbol{a}$ and $\boldsymbol{x}$ has as few 1s as possible. This problem is as hard to approximate as the smallest binary sub-decomposition when $\boxplus = \oplus$.

Proposition 9. *If $\boxplus = \oplus$, the smallest binary sub-decomposition problem is as hard to approximate as the minimum weight codeword problem.*

Proof. Notice first that we can replace the all-zeros vector $\boldsymbol{a}$ in the minimum weight codeword problem by an arbitrary binary vector $\boldsymbol{c}$ by adding that vector

also as a column to $\boldsymbol{B}$ and by adding a new row to $\boldsymbol{B}$ and $\boldsymbol{c}$ to enforce that $\boldsymbol{c}$ must be part of the solution, similarly to the proof of Proposition 5.

To see that the smallest binary sub-decomposition is at least as hard to approximate as the minimum weight codeword, notice that the reduction in Proposition 5 is approximation-preserving.

To see that the smallest binary sub-decomposition is no harder to approximate than the minimum weight codeword, consider an instance $(\boldsymbol{A}, D = \{\boldsymbol{F}_1, \ldots, \boldsymbol{F}_k\})$ of the sub-decomposition problem. Re-shape matrix $\boldsymbol{A}$ into a nm-dimensional binary column vector $\boldsymbol{a}$. Reshape all matrices $\boldsymbol{F}_k$ similarly into binary vectors $\boldsymbol{f}_k$, and collect them as columns of nm-by-$|D|$ binary matrix $\boldsymbol{B}$. Our task now is to select the least number of columns of $\boldsymbol{B}$ such that their sum modulo-2 is $\boldsymbol{a}$, that is, to find the vector $\boldsymbol{x}$ in the minimum-weight codeword problem. This reduction is also approximation-preserving, concluding the proof. □

For the following results, we need the concept of *(randomized) quasi*-NP-*hardness*.

Definition 11. *We say a problem Π is* quasi-NP-hard *if Π cannot be solved in polynomial time unless* NP $\subseteq$ DTIME($n^{\text{polylog}(n)}$). *We say Π is* randomized quasi-NP-hard*, if it cannot be solved in polynomial time unless* NP $\subseteq$ RTIME($n^{\text{polylog}(n)}$)*, where* RTIME($n^{\text{polylog}(n)}$) *is the set of all languages recognizable by Monte Carlo algorithms with probability exceeding* 1/2 *in time* $O(n^{\text{polylog}(n)})$.

Corollary 3. *If $\boxplus = \oplus$, the smallest binary sub-decomposition is randomized quasi*-NP-*hard to approximate to within $2^{\log^{1-\varepsilon} k}$ for any $\varepsilon > 0$. It can, however, be approximated in polynomial time to within εk for any fixed ε.*

Proof. The negative result follows from a result of Dumer et al. [11] and Proposition 9. The positive result comes from Berman and Karpinski [8] and Proposition 9. □

5.2 Approximating Minimum-Error Sub-Decompositions

We now turn our attention to the minimum-error sub-decompositions.

Proposition 10. *Let $(\boldsymbol{A}, D = \{\boldsymbol{F}_i\})$, with $|D| = k$, be an input for the minimum-error binary sub-decomposition problem. If $\boxplus = \vee$, it is quasi*-NP-*hard to approximate the minimum-error binary sub-decomposition problem to within a factor of $\Omega(2^{(4 \log k)^{1-\varepsilon}})$ and* NP-*hard to approximate it within $\Omega(2^{\log^{1-\varepsilon}|\boldsymbol{A}|})$ for any $\varepsilon > 0$. The problem can be approximated to within a factor of $2\sqrt{(k + |\boldsymbol{A}|) \log |\boldsymbol{A}|}$ in polynomial time.*

Proof. The hardness-of-approximation result follows directly from the proof of Proposition 6 in case of $\boxplus = \vee$: the reduction from the basis usage problem is approximation-preserving, and corresponding lower bounds are known for the basis usage problem [24, 25].

For the positive result, we need the reduction to go the other way. Let $(\boldsymbol{A}, D = \{\boldsymbol{F}_i\})$ be an input for the minimum-error sub-decomposition problem. Re-shape $\boldsymbol{A}$ to an nm-dimensional column vector $\boldsymbol{a}$, and re-shape factor matrices $\boldsymbol{F}_i$ similarly and collect them into an nm-by-k binary matrix $\boldsymbol{B}$. This is a valid input for the basis usage problem, and the solution can be approximated using Peleg's algorithm [30]. The result can be mapped back to minimum-error sub-decomposition problem without any change in the target value, and hence the reduction is approximation-preserving. The claim follows from known upper bounds for Peleg's algorithm for basis usage problem [25]. □

It is interesting to notice that the size of $\boldsymbol{A}$ plays no role in Proposition 10, only the number of its non-zeros. Also, quasi-NP-hardness is slightly stronger assumption than randomized quasi-NP-hardness that was assumed in Proposition 9.

The claim (and proof) for $\boxplus = \oplus$ is similar to the above.

Proposition 11. *Let $(\boldsymbol{A} \in \{0,1\}^{n \times m}, D = \{\boldsymbol{F}_i\})$, with $|D| = k$, be an input for the minimum-error binary sub-decomposition problem. If $\boxplus = \oplus$, it is quasi-*NP*-hard to approximate the minimum-error binary sub-decomposition problem to within a factor of $\Omega(2^{\log^{0.8-\varepsilon} n})$ for any $\varepsilon > 0$ and* NP*-hard to approximate it to within any constant factor. The problem can be approximated to within a factor of $O(k/\log(nm))$ in randomized polynomial time and to the same factor deterministically in time $(nm)^{O(\log^* nm)}$.*

Proof. This proof is similar to the above ones, and only a sketch of the proof is presented. The reduction from the nearest codeword problem in the proof of Proposition 6 is approximation-preserving, giving the negative result when paired with the results from [4]. The positive results require similar re-writing as above, after which we can use the results from [8] and [1]. □

6 Conclusions and Future Work

This paper presents an approach to unify pattern set mining using generalized outer products. Not every type of pattern can be expressed as a generalized outer product – not, at least, without making the outer products so general that we lose any reasonable way to study them as a group. Yet, as the above discussion has demonstrated, many interesting types of patterns – stars, biclique covers, nested matrices, and others – can easily be expressed in the framework, making it probable that also many yet-to-be-invented types of patterns will fit into it.

When a new type of pattern set mining problem can be expressed within the proposed framework, the researcher gains many benefits: the connections to other, sometimes seemingly unrelated work became more clear, many existing results might already apply, saving the researcher from tedious proofs, and the new results and techniques could generalize as well, immediately benefitting the whole field.

The work presented in this paper is only the beginning. The results concentrate on binary matrices, but recent work has generalized the binary setting to ordered lattices [5], ternary values [23], and rank matrices [20]. The general framework presented here could be extended to these situations, as well.

Another line of research extending the framework is to move from matrices to tensors (i.e. multi-way arrays). Again, there exists precedence in the pattern set mining, where mining higher-order (binary) data has gathered significant research interest [7, 9, 26].

Instead of finding the smallest or minimum-error pattern set, one can seek for a *planted* pattern, that is, a pattern we know the data contains, but that has been perturbed by noise. Recent research has shown that we can find individual planted patterns relatively well, even under strong noise assumptions [2, 31].

There is, then, a lot to do to before the proposed framework can gain its full power. Yet, even this short preliminary work should be enough to demonstrate the potential the framework has, and – hopefully – convince researchers to frame their research within it.

References

1. Alon, N., Panigrahy, R., Yekhanin, S.: Deterministic Approximation Algorithms for the Nearest Codeword Problem. In: APPROX RANDOM '09. pp. 339–351. Springer Berlin Heidelberg (2009)
2. Ames, B.P.W., Vavasis, S.A.: Nuclear norm minimization for the planted clique and biclique problems. Math. Program. B 129(1), 69–89 (2011)
3. Araujo, M., Günnemann, S., Mateos, G., Faloutsos, C.: Beyond Blocks: Hyperbolic Community Detection. In: ECML PKDD '14. pp. 50–65 (2014)
4. Arora, S., Babai, L., Stern, J., Sweedyk, Z.: The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. In: FOCS '93. pp. 724–733 (1993)
5. Bělohlávek, R., Krmelova, M.: Beyond Boolean Matrix Decompositions: Toward Factor Analysis and Dimensionality Reduction of Ordinal Data. In: ICDM '13. pp. 961–966 (2013)
6. Bělohlávek, R., Vychodil, V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. J. Comput. Syst. Sci. 76(1), 3–20 (2010)
7. Bělohlávek, R., Vychodil, V.: Factorizing Three-Way Binary Data with Triadic Formal Concepts. In: KES '10. pp. 471–480 (2010)
8. Berman, P., Karpinski, M.: Approximating minimum unsatisfiability of linear equations. In: SODA '02. pp. 514–516 (2002)
9. Cerf, L., Besson, J., Nguyen, K.N.T., Boulicaut, J.F.: Closed and noise-tolerant patterns in n-ary relations. Data Min. Knowl. Discov. 26(3), 574–619 (2013)
10. De Bie, T.: Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Min. Knowl. Discov. 23(3), 407–446 (2011)
11. Dumer, I., Micciancio, D., Sudan, M.: Hardness of approximating the minimum distance of a linear code. IEEE Trans. Inform. Theory 49(1), 22–37 (Jan 2003)
12. Ene, A., Horne, W., Milosavljevic, N., Rao, P., Schreiber, R., Tarjan, R.E.: Fast exact and heuristic methods for role minimization problems. In: SACMAT '08. pp. 1–10 (2008)

13. Feige, U.: A threshold of $\ln n$ for Approximating Set Cover. J. ACM 45(4), 634–652 (1998)
14. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-Completeness. W. H. Freeman, New York (1979)
15. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: DS '04. pp. 278–289 (2004)
16. Johnson, D.S.: Approximation Algorithms for Combinatorial Problems. J. Comput. Syst. Sci. 9, 256–278 (1974)
17. Junttila, E.: Patterns in permuted binary matrices. Ph.D. thesis, Helsinki University Press, Helsinki (Aug 2011)
18. Kötter, T., Günnemann, S., Berthold, M., Faloutsos, C.: Extracting Taxonomies from Bipartite Graphs. In: WWW '15 Companion. pp. 51–52 (2015)
19. Koutra, D., Kang, U., Vreeken, J., Faloutsos, C.: VoG: Summarizing and Understanding Large Graphs. In: SDM '14. pp. 91–99 (2014)
20. Le Van, T., van Leeuwen, M., Nijssen, S., Fierro, A.C., Marchal, K., De Raedt, L.: Ranked Tiling. In: ECML PKDD '14. pp. 98–113 (2014)
21. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. J. Comput. Syst. Sci. 20(2), 219–230 (1980)
22. Lucchese, C., Orlando, S., Perego, R.: A Unifying Framework for Mining Approximate Top-k Binary Patterns. IEEE Trans. Knowl. Data Eng. 26(12), 2900–2913 (2013)
23. Maurus, S., Plant, C.: Ternary Matrix Factorization. In: ICDM '14. pp. 400–409 (2014)
24. Miettinen, P.: On the positive-negative partial set cover problem. Inform. Process. Lett. 108(4), 219–221 (2008)
25. Miettinen, P.: Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms. Ph.D. thesis, Department of Computer Science, University of Helsinki (2009)
26. Miettinen, P.: Boolean Tensor Factorizations. In: ICDM '11. pp. 447–456 (2011)
27. Miettinen, P.: Fully dynamic quasi-biclique edge covers via Boolean matrix factorizations. In: DyNetMM '13. pp. 17–24 (2013)
28. Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., Mannila, H.: The Discrete Basis Problem. IEEE Trans. Knowl. Data Eng. 20(10), 1348–1362 (2008)
29. Peeters, R.: The maximum edge biclique problem is NP-complete. Discrete Appl. Math. 131(3), 651–654 (2003)
30. Peleg, D.: Approximation algorithms for the Label-Cover$_{MAX}$ and Red-Blue Set Cover problems. J. Discrete Alg. 5(1), 55–64 (2007)
31. Ramon, J., Miettinen, P., Vreeken, J.: Detecting Bicliques in GF[q]. In: ECML PKDD '13. pp. 509–524 (2013)
32. Simon, H.U.: On approximate solutions for combinatorial optimization problems. SIAM J. Discrete Math. 3(2), 294–310 (1990)
33. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Min. Knowl. Discov. 23(1), 169–214 (2011)
34. Xiang, Y., Jin, R., Fuhry, D., Dragan, F.F.: Summarizing transactional databases with overlapped hyperrectangles. Data Min. Knowl. Discov. 23(2), 215–251 (2011)
35. Yang, J., Leskovec, J.: Overlapping community detection at scale: a nonnegative matrix factorization approach. In: WSDM '13 (2013)
36. Yannakakis, M.: Node-Deletion Problems on Bipartite Graphs. SIAM J. Comput. 10(2), 310–327 (1981)