Revisiting the GreCon Algorithm for Boolean Matrix Factorization

Martin Trnecka[0000−0001−7770−2033] and Roman Vyjidacek[0000−0002−5442−0444]

Dept. Computer Science, Palacký University Olomouc, Olomouc, Czech Republic
martin.trnecka@gmail.com, roman.vyjidacek@upol.cz

Abstract. Over the past decade, the two most fundamental Boolean matrix factorization (BMF) algorithms, GreCon and GreConD, were proposed. Whereas GreConD has become one of the most popular algorithms, GreCon—an algorithm on which the GreConD was build—is somewhat forgotten. Although GreCon may produce better results than GreConD, computing BMF via this algorithm is time consuming. We show that the reasons for not using GreCon algorithm are no longer truth. We revise the algorithm and on various experiments we demonstrate that the revised version is competitive to current BMF algorithms in term of running time. Moreover, in some cases GreCon outperforms GreConD—the fastest BMF algorithm. Additionally, we argue that a search strategy of GreConD, notwithstanding it provides a good result, is limited. Furthermore, we show that our novel approach to GreCon opens a new door to further BMF research.

Keywords: Boolean matrix factorization · Boolean matrix factorization Algorithms · Formal concept analysis

CEUR-WS.org/Vol-2668/paper4.pdf

1 Introduction

Boolean Matrix Factorization (BMF), also known as Boolean matrix decomposition, is a well-established and widely used tool in data-mining and data processing of Boolean (1/0) data.

In the last decade there was a huge effort dedicated to a BMF research. In many works (e.g. [2, 3]) a strong connection—which we consider in the paper—between BMF and formal concept analysis (FCA) was established. FCA provides a general framework for the BMF problem description. Namely, formal context represents the input data and formal concepts represent factors in such data. From this perspective, BMF can be seen as a covering of a formal context by formal concepts [2, 3, 10].

In the pioneer work [3] two main algorithms, GreCon and GreConD, for BMF as well as a fundamental theory based on FCA were established.[1] These two algorithms are basic ones. Both utilize formal concepts as candidates for

[1] The algorithms were originally called Algorithm 1 and 2. The names GreCon and GreConD come from later works.

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0). Published in Francisco J. Valverde-Albacete, Martin Trnecka (Eds.): Proceedings of the 15th International Conference on Concept Lattices and Their Applications, CLA 2020, pp. 59–70, 2020.

factors. From these candidates the final factors are selected via a greedy choice. A detailed description of these two algorithms is provided in Section 2.

While the GreConD has become extremely popular, mainly due to its simple architecture and performance—in a fact GreConD is one of the fastest BMF algorithms and according to various experimental evaluations (see e.g. results in [1, 2]) it provides a very good results from the quality viewpoint—the GreCon is forgotten in the contemporary BMF research. There are two main reasons for that: (i) GreCon is a very slow BMF algorithm. The algorithm requires a several iterations over the set of all formal concepts, which may be large. (ii) Results provided by GreConD are comparable to results provided by GreCon. These two reasons stand on results presented in [3] and are widely adopted in BMF research. In the paper, we argue that these reasons need to be revisited, mainly according to a new development in BMF.

The main aim of this paper is to show that the reasons for not using GreCon algorithm are no longer valid—thanks to the development of algorithms for FCA and, paradoxically, thanks to the development of the GreConD algorithm. The main aim can be summarized as follows: (i) We reimplemented the GreCon algorithm and on various experiments we demonstrate that the revised version is competitive in term of running time with existing BMF algorithms. Moreover, the new implementation in some cases outperforms GreConD. (ii) We show, that the search strategy used by GreConD, despite it provides a good result, is limited. In the paper, we compare the search spaces of GreCon and GreConD and we briefly address the comparison of the results provided by both of them. (iii) Last but not least, the massive speed up of GreCon presented in the paper opens a new door to further BMF research. Namely, we utilize the GreCon search strategy on a restricted set of formal concepts. Although this is not a completely new idea, its application was limited, because of the speed of existing algorithms. Moreover, in an experimental evaluation we show that this approach may provides better results.

The rest of the paper is organized as follows. In Section 2, we provide a brief introduction to BMF, overview of related works and a description of the basic algorithms, GreCon and GreConD. Then, in Section 3, a comparison of GreCon and GreConD is discussed. Section 4 provides a description and a pseudocode of the redesign GreCon algorithm. In Section 5, we present results from various experimental evaluations. Section 6 summarizes the paper and outlines a further research directions.

2 A Brief Introduction to BMF

A good overview of BMF and related topics can be found e.g. in [1, 2, 8]. In general, BMF and BMF algorithms are addressed in various papers involving formal concept analysis [3, 6], role mining [4], binary databases [5] or bipartite graphs [9]. In this paper, we focus instead of a general BMF to a certain class of factorization, so-called *from-below matrix factorization* [2], i.e. no entries in the input data with a zero value are covered by some factor.

A general aim of BMF is for a given Boolean matrix $\mathbf{I} \in \{0,1\}^{m\times n}$ to find matrices $\mathbf{A} \in \{0,1\}^{m\times k}$ and $\mathbf{B} \in \{0,1\}^{k\times n}$ for which

$$\mathbf{I} \approx \mathbf{A} \circ \mathbf{B} \tag{1}$$

where $\circ$ is Boolean matrix multiplication, i.e. $(\mathbf{A} \circ \mathbf{B})_{ij} = \max_{l=1}^{k} \min(\mathbf{A}_{il}, \mathbf{B}_{lj})$, and $\approx$ represents an approximate equality. This approximate equality is assessed by $|| \cdot ||$ (i.e. by number of 1s) and with the corresponding metric E which is defined for matrices $\mathbf{I} \in \{0,1\}^{m\times n}$, $\mathbf{A} \in \{0,1\}^{m\times k}$, and $\mathbf{B} \in \{0,1\}^{k\times n}$ by

$$E(\mathbf{I}, \mathbf{A} \circ \mathbf{B}) = ||\mathbf{I} \ominus (\mathbf{A} \circ \mathbf{B})||, \tag{2}$$

where $\ominus$ is *Boolean subtraction* which is the normal matrix subtraction with an alternative definition $0 - 1 = 0$. In other words, function E is a number of 1s in $\mathbf{I}$ that are not in $(\mathbf{A} \circ \mathbf{B})$. The metric (2), or its variant, is generally used to assess the quality of factorization [1,2].

A decomposition of $\mathbf{I}$ into $\mathbf{A} \circ \mathbf{B}$ may be interpreted as a discovery of k factors that exactly or approximately describe the data: interpreting $\mathbf{I}$, $\mathbf{A}$, and $\mathbf{B}$ as the object-attribute, object-factor, and factor-attribute matrices. The model (1) can be interpreted as follows: the object i has the attribute j, i.e. $\mathbf{I}_{ij} = 1$, if and only if there exists factor l such that l applies to i and j is one of the particular manifestations of l.

2.1 BMF with Help of Formal Concept Analysis

We already mentioned that the BMF is closely connected to FCA. The formal context $\langle X, Y, I\rangle$ with m objects and n attributes can be seen as a Boolean matrix $\mathbf{I} \in \{0,1\}^{m\times n}$ where $\mathbf{I}_{ij} = 1$ if $\langle x, y\rangle \in I$, and vice versa. To every $\mathbf{I} \in \{0,1\}^{n\times m}$ one may associate a pair $\langle {}^{\uparrow}, {}^{\downarrow}\rangle$ of arrow operators assigning to sets $C \subseteq X = \{1, \ldots, m\}$ and $D \subseteq Y = \{1, \ldots, n\}$ the sets $C^{\uparrow} \subseteq Y$ and $D^{\downarrow} \subseteq X$ defined by

$$C^{\uparrow} = \{j \in Y \mid \forall i \in C : \mathbf{I}_{ij} = 1\},$$
$$D^{\downarrow} = \{i \in X \mid \forall j \in D : \mathbf{I}_{ij} = 1\}.$$

A pair $\langle C, D\rangle$ for which $C^{\uparrow} = D$ and $D^{\downarrow} = C$ is called a *formal concept*. The set of all formal concepts for formal context $\langle X, Y, I\rangle$ is defined as follows

$$\mathcal{B}(X, Y, I) = \{\langle C, D\rangle \mid C \subseteq X, D \subseteq Y, C^{\uparrow} = D, D^{\downarrow} = C\}.$$

For the sake of simplicity, we denote the set of all formal concepts for Boolean matrix $\mathbf{I}$ by $\mathcal{B}(\mathbf{I})$. The set of all concepts can be equipped with a partial order $\leq$ such that $\langle A, B\rangle \leq \langle C, D\rangle$ iff $A \subseteq C$ (or $D \subseteq B$). The whole set of partially ordered formal concepts is called the *concept lattice* of $\mathbf{I}$.

The set of formal concepts that is generated by single object (denoted by $\mathcal{O}(\mathbf{I})$) is called the set of object concepts, i.e. $\mathcal{O}(\mathbf{I}) = \{\langle i^{\uparrow\downarrow}, i^{\uparrow}\rangle \mid \forall i \in X\}$, and the set that is generated by single attribute (denoted $\mathcal{A}(\mathbf{I})$) is called the set of attributes concepts, i.e. $\mathcal{A}(\mathbf{I}) = \{\langle j^{\downarrow}, j^{\downarrow\uparrow}\rangle \mid \forall j \in Y\}$.

Now, we explain the connection between a set of formal concepts and the BMF. Every set $\mathcal{F} = \{\langle C_1, D_1\rangle, \ldots, \langle C_k, D_k\rangle\} \subseteq \mathcal{B}(\mathbf{I})$, with a fixed indexing of the formal concepts $\langle C_l, D_l\rangle$ induces the $m \times k$ and $k \times n$ Boolean matrices $\mathbf{A}_{\mathcal{F}}$ and $\mathbf{B}_{\mathcal{F}}$ by

$$(\mathbf{A}_{\mathcal{F}})_{il} = \begin{cases} 1, \text{if } i \in C_l, \\ 0, \text{if } i \notin C_l, \end{cases}$$

and

$$(\mathbf{B}_{\mathcal{F}})_{lj} = \begin{cases} 1, \text{if } j \in D_l, \\ 0, \text{if } j \notin D_l, \end{cases}$$

for $l = 1, \ldots, k$. That is, the lth column of $\mathbf{A}_{\mathcal{F}}$ and lth row $\mathbf{B}_{\mathcal{F}}$ are the characteristic vectors of C_l and D_l, respectively. The set $\mathcal{F}$ is called a set of *factor concepts*. The entry $\mathbf{I}_{ij} = 1$ is *covered* by formal concept $\langle A, B\rangle$ if $i \in A$ and $j \in B$.

2.2 Algorithm GreCon

The GreCon[2] algorithm [3] is one of the straightforward algorithms for BMF based on FCA. To produce matrices $\mathbf{A}_{\mathcal{F}}$ and $\mathbf{B}_{\mathcal{F}}$ it uses a greedy search for factor concepts driven by metric (2). More precisely, the algorithm first computes the set $\mathcal{B}(\mathbf{I})$. Then it iteratively goes through this set and in each iteration a factor concept that covers the largest not yet covered part of input data is selected, i.e. it is a set cover based algorithm. The algorithm is designed to compute an exact (or approximate if it is stopped before the end) from-below matrix factorization.

2.3 Algorithm GreConD

One of the most successful algorithms for BMF is the GreConD[3] algorithm [3] which was originally designed to improve a running time of GreCon. To produce the matrices $\mathbf{A}_{\mathcal{F}}$ and $\mathbf{B}_{\mathcal{F}}$ it uses a particular greedy search for factor concepts which allows us to compute factor concepts "on demand", i.e. without the need to compute the set of all formal concepts for the input matrix first. GreConD constructs the factor concepts by adding sequentially "promising columns" to candidate $\langle C, D\rangle$ for factor concept. More formally, a new column j that minimizes the error $E(\mathbf{I}, \mathbf{A}_{\mathcal{F}\cup\langle(D\cup j)^{\downarrow},(D\cup j)^{\downarrow\uparrow}\rangle} \circ \mathbf{B}_{\mathcal{F}\cup\langle(D\cup j)^{\downarrow},(D\cup j)^{\downarrow\uparrow}\rangle})$ is added to $\langle C, D\rangle$. This is repeated until no such columns exist. If there is no such column, $\langle C, D\rangle$ is added to the set $\mathcal{F}$. This strategy leads to a huge time saving while the quality of the decomposition is comparable with the decomposition obtained via GreCon. Similar to GreCon, the algorithm is also designed to compute an exact and approximate from-below factorization.

[2] GreCon is the abbreviation for Greedy Concepts.
[3] GreConD is the abbreviation for Greedy Concepts on Demand.

3 Comparison of GreCon and GreConD

In what follows, we demonstrate differences between the search strategies of GreConD and GreCon. Let us consider matrix $\mathbf{I}$, depicted below, with rows $\{1, 2, 3, 4, 5\}$ and columns $\{a, b, c, d, e, f\}$

$$\mathbf{I} = \begin{array}{c c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} & \\ \begin{pmatrix} 0&0&1&1&1&1 \\ 0&1&0&1&0&1 \\ 1&1&0&0&1&1 \\ 1&0&0&1&1&0 \\ 0&0&0&1&1&0 \end{pmatrix} & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \end{array}$$

and two sets of factor concepts $\mathcal{F}_1$ and $\mathcal{F}_2$ that are computed via GreConD and GreCon respectively. The corresponding matrices are:

$$\mathbf{A}_{\mathcal{F}_1} \circ \mathbf{B}_{\mathcal{F}_1} = \begin{pmatrix} 0&0&1&1&1 \\ 0&1&0&1&0 \\ 1&1&0&0&1 \\ 1&0&0&1&1 \\ 0&0&0&1&1 \end{pmatrix} \circ \begin{pmatrix} 1&0&0&0&1&0 \\ 0&1&0&0&0&1 \\ 0&0&1&1&1&1 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \end{pmatrix}, \mathbf{A}_{\mathcal{F}_2} \circ \mathbf{B}_{\mathcal{F}_2} = \begin{pmatrix} 1&0&0&1&0 \\ 0&0&1&0&0 \\ 0&1&0&0&1 \\ 1&0&0&0&1 \\ 1&0&0&0&0 \end{pmatrix} \circ \begin{pmatrix} 0&0&0&1&1&0 \\ 1&1&0&0&1&1 \\ 0&1&0&1&0&1 \\ 0&0&1&1&1&1 \\ 1&0&0&0&1&0 \end{pmatrix}.$$

The search space, in each iteration of GreCon, is the set of all formal concepts. Namely, in each the iteration all formal concepts in the concept lattice of $\mathbf{I}$ (depicted in Figure 1) are considered as candidates for factors, in order in they were generated.

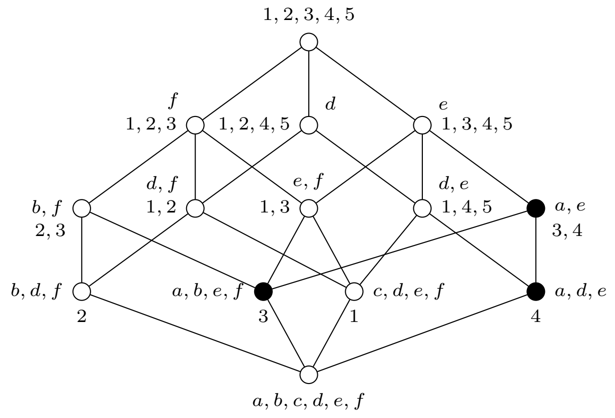


Fig. 1: Concept lattice of $\mathbf{I}$.

The search space of GreConD is different. GreConD always starts with attributes concepts. Then it extends the selected concept via some promising attributes, i.e. GreConD search space is limited to chains in the corresponding concept lattice and the selection of a particular chain is driven by the greedy choice. Note, the columns of the input matrix are considered in a fixed order. As a consequence of this GreConD may stop in some local maximum.

In a fact, this is true for the first iteration on our example data matrix $\mathbf{I}$. GreConD starts with attribute a which generates $\langle\{3,4\},\{a,e\}\rangle$. All remaining candidates (attribute concepts) have smaller or equal size as the first concept. Then, $\langle\{3,4\},\{a,e\}\rangle$ is extended by some attributes to obtain a potentially better candidate (candidate which covers more not yet covered entries of $\mathbf{I}$), namely attributes b,d,f are considered, meanwhile c is skipped because a,c,e do not generate any formal concept. The concepts considered as candidates for factors in the first iteration of GreConD are depicted by black nodes in Figure 1. As a consequence of this GreConD is not able to discover the best choice $\langle\{1,4,5\},\{d,e\}\rangle$.

Despite the differences between the algorithms, the results produced by them are comparable—GreConD produces slightly worse results than GreCon—in terms of the overall coverage which is evaluated by the metric (2). The comparison of the coverage is shown in Figure 2. Note, the coverage is computed as $1 - \frac{E(\mathbf{I},\mathbf{A}\circ\mathbf{B})}{||I||}$.

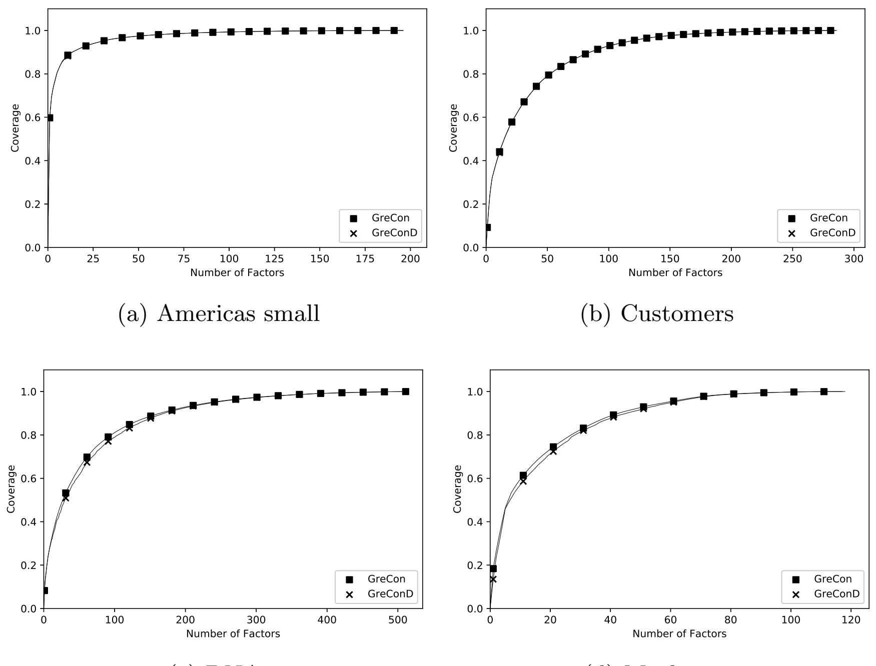
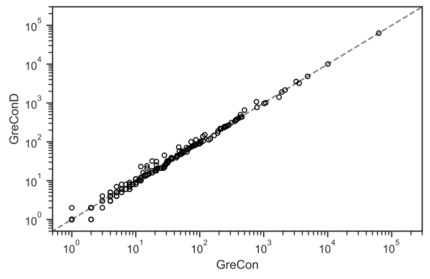


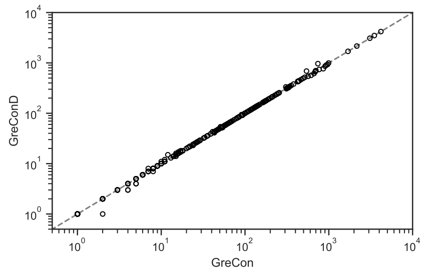
Fig. 2: Comparison of coverage produced by factorization obtained via GreCon and GreConD on selected real-word data.

This observation—in fact results presented in Figure 2 are standard in BMF—was confirmed by previous works e.g. [1, 2]. The interpretation of this observation is affected by a small misunderstanding. Namely, graphs in Figure 2 capture the cumulative coverage, i.e. the sum of all covered entries for a given number of factors. In this sum, the differences between the factorizations be may not easy to see.

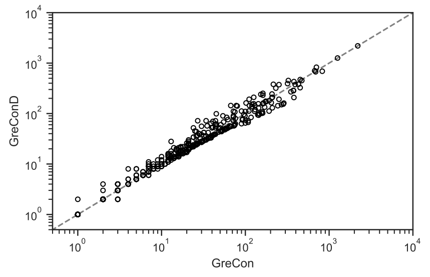
In [3] a different kind of experiments were performed. Instead of plotting the cumulative coverage, the coverage of i-th factor in one factorization in contrast with the coverage of i-th factor in the second factorization is plotted, i.e. plotted points have coordinates based on the number of covered entries. Results of this experiments—[3] includes only the results for Mushroom data (see Section 5)—for selected data are depicted in Figure 3.



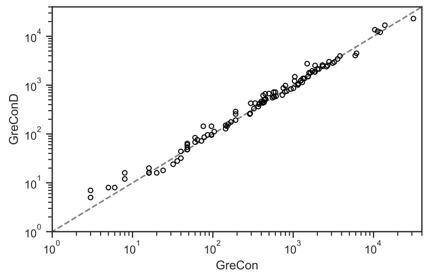
(a) Americas small



(b) Customers



(c) DNA



(d) Mushroom

Fig. 3: Comparison of GreCon and GreConD algorithms on selected real-word data. The plot uses axes with a logarithmic scale. The plotted points have coordinates based on the number of covered entries.

One may easily see, that the difference between factorizations may be very large despite small differences in Figure 2. In case of Customers dataset (Figure 3b) the datapoints are gathered close to the diagonal which means that there is a small difference between the corresponding factors. In contrast with this a significant difference between GreCon and GreConD on DNA dataset (Figure 3c) may be observed.

4 Redesign of GreCon

Now we describe a basic idea of redesigning the GreCon algorithm. We call the redesigned version of the algorithm GreCon2 because it is mainly an efficient implementation of the original GreCon algorithm.

Differently from GreCon, GreCon2 does not need repeatedly compute the number of not yet covered entries for each formal concept—which is the most time consuming task in GreCon. Instead of this the actual cover for all formal concepts is maintained for the entire running time. For this purpose, a convenient data structure which enables an additional speed up is used. Additionally, GreCon2 does not iterate over all the formal concepts (i.e. data structure) but only over the numbers (integers). A pseudocode of GreCon2, which accepts as an input Boolean matrix $\mathbf{I} \in \{0,1\}^{m \times n}$, is depicted in Algorithm 1.

Algorithm 1: GreCon2

```
Input: Boolean matrix $\mathbf{I} \in \{0,1\}^{m \times n}$
Output: Set $\mathcal{F}$ of factor concepts.

1  $concepts \leftarrow \mathcal{B}(\mathbf{I})$;
2  foreach $\langle A_l, B_l \rangle \in concepts$ do
3      $covers[l] \leftarrow ||A_l|| \cdot ||B_l||$
4      foreach $(i, j) \in \langle A_l, B_l \rangle$ do
5          append $l$ to list $cell[i \cdot n + j]$
6      end
7  end
8  while $\mathbf{A}_{\mathcal{F}} \circ \mathbf{B}_{\mathcal{F}} \neq \mathbf{I}$ do
9      add $\langle A_l, B_l \rangle$ which maximizes $covers[l]$ to $\mathcal{F}$
10     foreach $(i, j) \in \langle A_l, B_l \rangle$ do
11         foreach $k \in cell[i \cdot n + j]$ do
12             $covers[k] \leftarrow covers[k] - 1$
13         end
14         delete list $cell[i \cdot n + j]$
15     end
16 end
```

In the first phase, GreCon2 computes the set of all formal concepts $\mathcal{B}(\mathbf{I})$ (line 1). Then the algorithm computes a coverage for each concept—stores in array *covers* (line 3)—and for each $\mathbf{I}_{ij}$ constructs a list of concepts that cover $\mathbf{I}_{ij}$ (lines 4–6). These lists are stored in array *cell* on a corresponding indexes (line 5). Note that accessing the entries of arrays *covers* and *cell* requires a constant time.

In the second phase, GreCon2 performs linear search in array *covers* and adds to $\mathcal{F}$ concept $\langle A_l, B_l \rangle$ for which the value of $cover[l]$ is maximal (line 9). For each $\mathbf{I}_{ij}$ covered by $\langle A_l, B_l \rangle$ GreCon2 decrease values in array *covers* for each concept that covers $\mathbf{I}_{ij}$ (lines 10–13) and deletes the list for $\mathbf{I}_{ij}$ (line 14). The second phase is repeated until all the entries of the input data matrix are covered or, equivalently, there is no $covers[l] > 0$.

5 Experimental Evaluation

In this section, we present results of an experimental comparison of GreCon, GreConD and GreCon2. We implemented[4] all three algorithms in the Swift programming language with the same level of optimization to make the comparison fair.

5.1 Datasets

We used eight real-world datasets, namely Advertisement, Mushroom, Tic Tac Toe from [7]; Americas small, Apj, Customer, Firewall 1 from [4]; and DNA [11]. The characteristics of the datasets are shown in Table 1. Specifically the number of objects, the number of attributes, the density of nonzero entries in data in percents, the number of concepts, and the number of object and attribute concepts. All of them are well known and widely used as benchmark datasets in BMF.

Table 1: Datasets and their characteristics.

dataset	# objects	# attributes	dens.	$\lvert\mathcal{B}(I)\rvert$	$\lvert\mathcal{O}(I) \cup \mathcal{A}(I)\rvert$
Advertisement	3279	1557	0.88	9192	2648
Americas small	3477	1587	1.91	2764	524
Apj	2044	1164	0.29	798	723
Customer	10961	277	1.50	47848	5806
DNA	4590	392	1.47	4483	1450
Firewall 1	365	709	12.35	317	152
Mushroom	8124	119	17.65	221525	8231
Tic Tac Toe	958	30	33.33	59505	988

5.2 Running Times Comparison

We compared the running times of GreCon, GreConD and GreCon2. All experiments were performed on an ordinal computer with 2.7GHz Quad-Core Intel Core i7 processor and 16GB of RAM. Each algorithm was run 5 times on a particular data and the average time as well the standard deviation (the numbers after ±) are reported in Table 2. Note, the running times do not involve the time required to load input data. In case of GreCon and GreCon2, the time required for the computation of all formal concepts, which is done via our implementation of FCbO algorithm [12], is included in each iteration.

Table 2 shows that in all cases GreCon2 significantly outperforms original GreCon algorithm. Moreover, in most cases GreCon2 outperforms GreConD. GreConD performs well on datasets where the number of attributes

[4] All implementations together with scripts that performs all presented experiments are available on the GitHub https://github.com/rvyjidacek/experiments-cla2020

Table 2: Comparison of running times of GreCon, GreCon2 and GreConD in seconds. The average time over five iterations as well as standard deviations (the numbers after $\pm$) are presented.

dataset	GreCon	GreCon2	GreConD
Advertisement	916.40 ± 9.80	$\mathbf{1.75 \pm 0.06}$	295.72 ± 1.38
Americas small	94.68 ± 0.17	$\mathbf{1.37 \pm 0.02}$	96.46 ± 0.61
Apj	16.70 ± 0.14	$\mathbf{0.21 \pm 0.00}$	59.52 ± 0.23
Customer	1179.33 ± 4.85	$\mathbf{3.23 \pm 0.02}$	11.95 ± 0.15
DNA	133.71 ± 2.21	$\mathbf{0.48 \pm 0.01}$	21.9 ± 0.50
Firewall 1	0.76 ± 0.02	$\mathbf{0.12 \pm 0.00}$	4.51 ± 0.08
Mushroom	1139.91 ± 15.37	31.14 ± 0.22	$\mathbf{3.99 \pm 0.03}$
Tic Tac Toe	5.69 ± 0.17	0.78 ± 0.01	$\mathbf{0.04 \pm 0.00}$

Table 3: Comparison of running times and the final number of factors of GreCon2 restricted to $\mathcal{O}(\mathbf{I}) \cup \mathcal{A}(\mathbf{I})$ and unrestricted GreConD.

dataset	GreCon2	# factors	GreConD	# factors
Advertisement	$\mathbf{0.38 \pm 0.00}$	**756**	295.72 ± 1.38	766
Americas small	$\mathbf{0.22 \pm 0.01}$	208	96.46 ± 0.61	**197**
Apj	$\mathbf{0.07 \pm 0.00}$	**463**	59.52 ± 0.23	464
Customer	$\mathbf{0.41 \pm 0.00}$	**281**	11.95 ± 0.15	286
DNA	$\mathbf{0.15 \pm 0.00}$	515	21.90 ± 0.50	**511**
Firewall 1	$\mathbf{0.05 \pm 0.00}$	**66**	4.51 ± 0.08	**66**
Mushrooms	$\mathbf{0.42 \pm 0.00}$	**103**	3.99 ± 0.03	118
Tic Tac Toe	$\mathbf{0.01 \pm 0.00}$	**29**	0.04 ± 0.00	32

is rather small. This is the reason why GreConD outperforms GreCon2 on Mushroom dataset which has a larger number of concepts and only a small number of attributes.

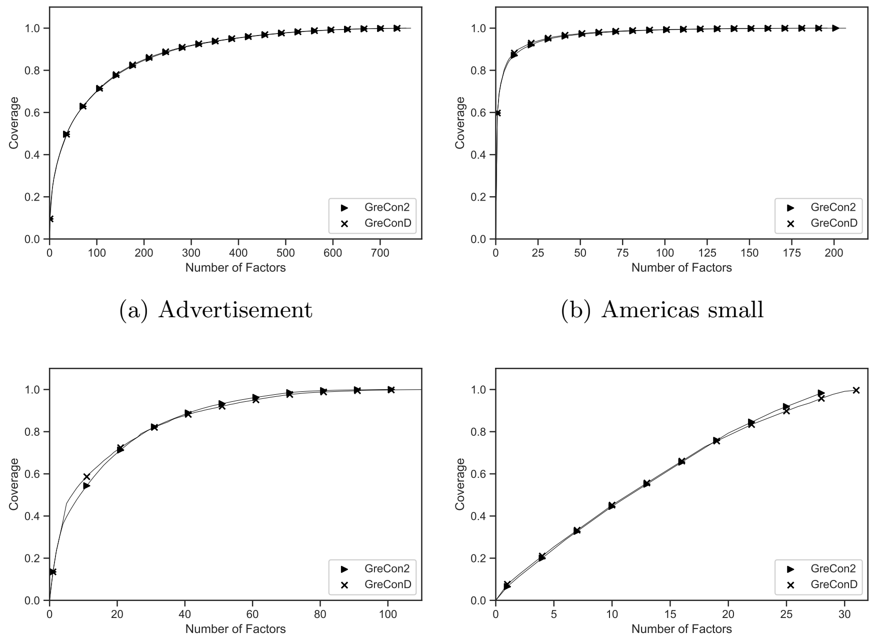
Note that our implementation of GreCon2 may be improved. Namely the construction phase of the algorithm (lines 2–6 in Algorithm 1) can be a part of FCbO algorithm (line 1). We decide to keep these two parts separated to make the comparison between GreCon and GreCon2 fair.

5.3 A New Approach to BMF

A significant speed up provided by GreCon2 allows us to consider the set of all formal concepts as a set of candidates for factors. Despite considerable speedup, the set may be still too large (like in the case of Mushroom dataset). As a very promising research direction seems to be a restriction of the set of formal concepts, i.e. skip formal concepts that are potentially not a good candidates for factors. To demonstrate this idea, we consider in GreCon2 the set $\mathcal{O}(\mathbf{I}) \cup$

$\mathcal{A}(\mathbf{I})$ instead of $\mathcal{B}(\mathbf{I})$. We perform the same experiments as above and compare GreCon2 with the restricted search space and unchanged GreConD.

From the results shown in Table 3, we observe that GreCon2 significantly outperforms GreConD in therm of running times. Moreover, in this case GreCon2 produces slightly better results in terms of overall coverage (see Figure 4) and in some cases a smaller number of factors than GreConD (see Table 3).



(a) Advertisement (b) Americas small

(c) Mushroom (d) Tic Tac Toe

Fig. 4: Comparison of coverage produced by factorization obtained via GreCon2 where $\mathcal{B}(\mathbf{I})$ was restricted to $\mathcal{O}(\mathbf{I}) \cup \mathcal{A}(\mathbf{I})$ and GreConD on selected datasets.

6 Conclusion and Further Research

We proposed a revised version of GreCon algorithm which significantly outperforms the original algorithm and which is competitive to the fastest BMF algorithms. Additionally, we show that our approach enables us to consider a larger search space than one of the most popular BMF algorithms, which in the past has caused that GreCon to be forgotten in BMF research.

Further research shall include an efficient implementation of the approach; a deep investigation of how the set of all formal concepts may be restricted without affecting the outcome quality; and last but not least, a parallelization of the approach.

Acknowledgments

Supported by Junior research Grant No. JG_2020_003 of the Palacký University Olomouc. Support by Grant No. IGA_PrF_2020_030 of IGA of Palacký University is also acknowledged.

References

1. Belohlavek, R., Outrata, J., Trnecka, M.: Toward quality assessment of Boolean matrix factorizations. Inf. Sci. **459**, 71–85 (2018). https://doi.org/10.1016/j.ins.2018.05.016
2. Belohlavek, R., Trnecka, M.: From-below approximations in Boolean matrix factorization: Geometry and new algorithm. J. Comput. Syst. Sci. **81**(8), 1678–1697 (2015). https://doi.org/10.1016/j.jcss.2015.06.002
3. Belohlavek, 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). https://doi.org/10.1016/j.jcss.2009.05.002
4. Ene, A., Horne, W.G., Milosavljevic, N., Rao, P., Schreiber, R., Tarjan, R.E.: Fast exact and heuristic methods for role minimization problems. In: Ray, I., Li, N. (eds.) 13th ACM Symposium on Access Control Models and Technologies, SACMAT 2008, Estes Park, CO, USA, June 11-13, 2008, Proceedings. pp. 1–10. ACM (2008). https://doi.org/10.1145/1377836.1377838
5. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: Suzuki, E., Arikawa, S. (eds.) Discovery Science, 7th International Conference, DS 2004, Padova, Italy, October 2-5, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3245, pp. 278–289. Springer (2004). https://doi.org/10.1007/978-3-540-30214-8_22
6. Ignatov, D.I., Nenova, E., Konstantinova, N., Konstantinov, A.V.: Boolean matrix factorisation for collaborative filtering: An fca-based approach. In: Agre, G., Hitzler, P., Krisnadhi, A.A., Kuznetsov, S.O. (eds.) Artificial Intelligence: Methodology, Systems, and Applications - 16th International Conference, AIMSA 2014, Varna, Bulgaria, September 11-13, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8722, pp. 47–58. Springer (2014). https://doi.org/10.1007/978-3-319-10554-3_5, https://doi.org/10.1007/978-3-319-10554-3_5
7. Lichman, M.: UCI machine learning repository (2013), http://archive.ics.uci.edu/ml
8. Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. IEEE Trans. Knowl. Data Eng. **20**(10), 1348–1362 (2008). https://doi.org/10.1109/TKDE.2008.53
9. Monson, S.D., Pullman, S., Rees, R.: A survey of clique and biclique coverings and factorizations of (0,1)-matrices. No. 14 (1995)
10. Mouakher, A., Yahia, S.B.: QualityCover: Efficient binary relation coverage guided by induced knowledge quality. Inf. Sci. **355-356**, 58–73 (2016). https://doi.org/10.1016/j.ins.2016.03.009
11. Myllykangas, S., Himberg, J., Böhling, T., Nagy, B., Hollmén, J., Knuutila, S.: Dna copy number amplification profiling of human neoplasms. Oncogene **25**(55), 7324–7332 (2006)
12. Outrata, J., Vychodil, V.: Fast algorithm for computing fixpoints of galois connections induced by object-attribute relational data. Inf. Sci. **185**(1), 114–127 (2012). https://doi.org/10.1016/j.ins.2011.09.023