The 8M Algorithm from Today's Perspective

Radim Belohlavek and Martin Trnecka

Palacky University, Olomouc, Czech Republic,
radim.belohlavek@acm.org, martin.trnecka@gmail.com

Abstract. 8M is an old but nowadays virtually unknown algorithm for Boolean matrix factorization. In this paper, we provide a detailed analysis of 8M. We demonstrate by experiments that even though the algorithm uses a limited insight into the decomposition problem, its performance is reasonably good even from today's perspective. We analyze all the steps involved in 8M, provide a first complete description of 8M, and the relationships of 8M to the main currently available factorization algorithms. It turns out that 8M involves certain interesting concepts, which are not exploited by the current algorithms. We discuss the prospect of these concepts and, furthermore, propose an enhancement of 8M which is based on the current understanding of Boolean matrix factorization and significantly improves the performance of the original 8M.

1 Introduction

1.1 The Goal of this Paper

In the past decade or so, considerable research has been devoted to Boolean matrix factorization (BMF, called also Boolean matrix decomposition). This research has resulted in various new methods of analysis and processing of Boolean data and has also contributed to our understanding of Boolean (binary, yes/no) data as regards foundational aspects. A vast majority of the respective research contributions has been devoted to the design of factorization algorithms, which is also the subject of our paper. To name some of the best-known algorithms (more detailed information about some of these algorithms is provided in the subsequent sections), let us recall Tiling [9], the nowadays classic Asso [13], GreConD [3], Hyper [19], PaNDa [11], GreEss [5], and various modifications of these algorithms and modifications of the factorization problems discussed in the above-mentioned papers, as well as in [4,10,12,14,16,18].

Interestingly, there exists an old BMF algorithm, namely the 8M algorithm, which is virtually unknown in the present research on BMF. This fact is remarkable particularly in view of our experimental evaluations which demonstrate that the 8M algorithm performs reasonably well even from today's perspective. We learned about this algorithm from Hana Řezanková who used it in her various works on comparison of various clustering and factorization methods; see e.g. the references in [2]. Even though the performance of 8M may be partially assessed from those works, the principles of 8M have never been discussed in

© Dmitry I. Ignatov, Lhouari Nourine (Eds.): CLA 2018, pp. 167–178,
ISBN 978–80–244–5328–6, Department of Computer Science, Palacký University
Olomouc, 2018.

the literature. The goal of this paper is threefold. First, we provide a complete description of the 8M algorithm, including its pseudo-code and the description of its principles from today's perspective. Second, we propose an improvement of the 8M algorithm, which turns out to improve its performance reasonably. Third, we utilize one of the principles of 8M to enhance the performance of two standard algorithms. Note at this point that we discussed the 8M algorithm in our yet unpublished paper [6] in which we were solely interested in one particular property of this algorithm which we exploited in [6]; the present description is complete and comprehensive compared to the one presented in [6].

1.2 Basic Notions

The set of all $n \times m$ Boolean matrices shall be denoted $\{0,1\}^{n\times m}$ and the particular matrices by I, J, and the like. An input matrix I shall primarily be interpreted as an object-attribute incidence matrix (hence the symbol I). That is, the entry I_{ij} corresponding to the row i and the column j is either 1 or 0, indicating that the object i does or does not have the attribute j, respectively. The ith row and jth column vector of I is denoted by $I_{i\_}$ and $I_{\_j}$, respectively. In BMF, one generally attempts to find for a given $I \in \{0,1\}^{n\times m}$ matrices $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$ for which

$$I \text{ (approximately) equals } A \circ B, \tag{1}$$

where $\circ$ is the Boolean matrix product, i.e. $(A \circ B)_{ij} = \max_{l=1}^{k} \min(A_{il}, B_{lj})$. A decomposition of I into $A \circ B$ may be interpreted as a discovery of k factors that exactly or approximately explain the data: Interpreting I, A, and B as object-attribute, object-factor, and factor-attribute matrices, model (1) reads: The object i has the attribute j if and only if there exists factor l such that l applies to i and j is one of the particular manifestations of l. The least k for which an exact decomposition $I = A \circ B$ exists is called the *Boolean rank* (or Schein rank) of I. The approximate equality in (1) is assessed in BMF by means of the metric $E(\cdot,\cdot)$, defined for $C, D \in \{0,1\}^{n\times m}$ by

$$E(C, D) = \textstyle\sum_{i,j=1}^{m,n} |C_{ij} - D_{ij}|. \tag{2}$$

The following particular variants of the BMF problem, relevant to this paper, are considered in the literature.

- *Discrete Basis Problem* (DBP, [13]):
  Given $I \in \{0,1\}^{n\times m}$ and a positive integer k, find $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$ that minimize $E(I, A \circ B)$.
- *Approximate Factorization Problem* (AFP, [3]):
  Given I and prescribed error $\varepsilon \geq 0$, find $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$ with k as small as possible such that $E(I, A \circ B) \leq \varepsilon$.

These problems reflect two important views of BMF: DBP emphasizes the importance of the first few (presumably most important) factors; AFP emphasizes the need to account for (and thus to explain) a prescribed portion of data.

In general, the committed error $E(I, A \circ B)$ has two parts, namely

$$E(I, A \circ B) = E_u(I, A \circ B) + E_o(I, A \circ B), \tag{3}$$

where $E_u(I, A \circ B) = |\{\langle i, j\rangle \mid I_{ij} = 1 \text{ and } (A \circ B)_{ij} = 0\}|$ and $E_o(I, A \circ B) = |\{\langle i, j\rangle \mid I_{ij} = 0 \text{ and } (A \circ B)_{ij} = 1\}|$ are the so-called undercovering error and overcovering error, respectively, which view shall be used below.

2 8M Described

2.1 History of 8M

The 8M method is one of the many data analysis methods available in an old and widely used statistical software package known as BMDP. The acronym "BMDP" stands for "Bio-Medical Data Package" (some sources say "BioMeDical Package"). The package was developed primarily for biomedical applications since the 1960s at the University of California in Los Angeles (UCLA) under the leadership of W. J. Dixon.[1] BMDP was originally available for free, later through BMDP Statistical Software, Inc., and then by its subsidiary, Statistical Solutions Ltd. As of 2017, BMDP is no longer available.[2]

BMDP and its methods are described in several editions of manuals, starting with a 1961 manual of BMD, a direct predecessor of BMDP. In our description of 8M, we use the 1992 edition [7], which accompanies release 7 of BMDP. There, 8M is described in chapter "Boolean factor analysis" on pp. 933–945, written by M. R. Mickey, L. Engelman, and P. Mundle, and in appendix B.11 on pp. 1401–1403.

The 8M method has been added to BMDP in the late 1970s: It was not part of the 1979 manual but it is part along with other new methods in the next version, whose revised printing appeared in 1983. According to this edition, 8M is based on research done by the statistician M. Ray Mickey of the UCLA, was designed by Mickey with contributions from Laszlo Engelman, and was programmed by Peter Mundle and Engelman.[3]

2.2 Description of the Method

Even though the description of 8M in [7] is fairly detailed, certain parts are somewhat unclear, both as to the procedural details and the rationale of various steps. As to the procedural details, we therefore examined the step-by-step

[1] The package grew out from an older computer program BIMED, which was developed for biomedical applications, and was first called BMD. Since the implemented methods allowed a parameterized format, the letter "P" was added. Later, "P" was interpreted as standing for "Package."

[2] We crosschecked our implementation against the version of BMDP we purchased in 2015 from Statistical Solutions Ltd.

[3] The references of the BMDP manual include some papers by Mickey but none of them concerns 8M and Boolean factor analysis.

program behavior on various data to figure out the unclear parts until our own implementation yielded the same results as the software which we purchased from Statistical Solutions Ltd. As to the rationale, we provide our explanation of the particular steps of 8M below.

Basic Idea We first describe the basic idea of 8M. The algorithm takes as its input four parameters: an $n \times m$ Boolean matrix I (object-attribute matrix), a number k of desired factors, and two auxiliary parameters, a number *init* of initial factors, and a number *cost* used to refine the factors being computed. The desired output consists of $n \times k$ and $k \times m$ Boolean matrices A (object-factor matrix) and B (factor-attribute matrix).

The algorithm starts by computing *init* initial factors. Then the algorithm iteratively computes new factors until k desired factors are obtained. The way 8M computes the factors is very different from the current BMF algorithms in two respects. First is the very way of generating a new factor. Second is the fact that the previously generated factors are revisited and dropped. The corresponding procedures are described in detail below.

Even though 8M's revisiting of the previously generated factors is done in a straightforward manner, it represents an interesting property. Namely, while the undercovering and overcovering error, E_u and E_o, see (3), seem symmetric, they have a different role in the design of BMF algorithms: Due to the NP-hardness of the various versions of the decomposition problem [17], most of the current factorization algorithms are heuristic approximation algorithms computing the factors one-by-one until a satisfactory factorization is obtained. Now, having computed say k factors, the next computed factor may make the overall error E smaller but its overcover part E_o never decreases (hence the decrease is E is due to a decrease in E_u). Put another way, while committing the E_u error may be repaired by adding further factors, committing the E_o error will never be repaired by adding further factors and must thus be carefully considered. Revisiting and possibly dropping some of the previously generated factors is a natural procedure to cope with this problem as it makes it possible to repair the E_o error. From this perspective it is interesting to note that while the current algorithms producing general factorization, such as Asso or PaNDa, do not use any kind of revisiting, the old 8M already used this idea.

Detailed Description and Pseudocode of 8M *(algorithm 1)* To compute $n \times k$ and $k \times m$ Boolean matrices A and B form the given $n \times m$ Boolean matrix I, the prescribed number *init* of initial factors, the desired number k of factors, and the parameter *cost*, the algorithm 8M (algorithm 1) proceeds as follows. First, *init* initial factors are computed (l. 1) as explained below. Note at this point that by default, $init = k - 2$ but *init* is generally set by the user. The variable f storing the number of the currently computed factors is set accordingly (l. 2). The matrices A and B are then refined (l. 3) by the procedure RefineMatricesAB described below. The algorithm then enters a loop (l. 5–17) whose purpose is to add new factors and remove some of the previously generated ones until the desired number k of factors is reached for the second time or all 1s in I are covered

Algorithm 1: 8M

Input: Boolean $n \times m$ matrix I, desired number of factors k, number *init* of initial factors, number *cost*
Output: Boolean matrices A and B

```
1  B ← ComputeInitialFactors(init); A ← 0_{n×init}
2  f ← init
3  RefineMatricesAB(A, B, I, cost)
4  kReached ← 0
5  while kReached < 2 or I ≤ A ∘ B do
6  |  foreach ⟨i, j⟩ do if I_ij > (A ∘ B)_ij then Δ⁺_ij ← 1 else Δ⁺_ij ← 0
7  |
8  |  add column j of Δ⁺ with the largest count of 1s as new column to A
9  |  add row of 0s as new row to B and set entry j of this row to 1
10 |  f ← f + 1
11 |  RefineMatricesAB(A, B, I, cost)
12 |  if another two new factors were added then
13 |  |  remove column A_{-(f-2)} from A and row B_{(f-2)-} from B
14 |  |  f ← f − 1
15 |  |  RefineMatricesAB(A, B, I, cost)
16 |  end
17 |  if f=k then kReached ← kReached + 1
18 end
19 return A, B
```

by $A \circ B$, i.e. $I_{ij} \leq (A \circ B)_{ij}$ for all i, j holds (l. 5). Whenever a factor is added or removed, A and B are refined. Adding and removing factors is performed according to the following scheme. One starts with $f = \mathit{init}$ factors, adds two factors so that $f + 2$ factors are obtained, then removes the factor generated two steps back, i.e. the fth factor, adds another two factors, removes a factor generated two steps back, and so on. Hence, starting with $\mathit{init} = 2$ factors, one successively obtains 2, 3, 4, 3, 4, 5, 4, 5, 6, 5, 6, 7, 6, 7, 8, etc. factors. One stops when the desired number k of factors is obtained the second time. For instance, if $k = 6$ one computes the sequence $2, 3, 4, 3, 4, 5, 4, 5, 6, 5, 6$ of factors and the last six factors are the final factors output by the algorithm (provided the algorithm does not stop due to the second condition in l. 5).

The initial factors are computed by ComputeInitialFactors (algorithm 2) as follows. First, an $m \times m$ matrix C is computed in which $C_{ij} = 1$ iff column i is included in column j in I (i.e. $I_{qi} \leq I_{qj}$ for each q). One then goes through the rows i of C, $i = 1, 2, \ldots$, and adds them as new rows of B until *init* rows have been added: row i of C is added to B provided there exists j with $C_{ij} = 1$ such that no row previously added to B contains 1 at position j.

Initialization of the factors is a key step in 8M in that the quality of the computed factorization depends on it. Below we propose a new way to initialize. At this point, let us point out an interesting observation. Computing the association matrix in the Asso algorithm is a kind of initialization. In particular,

Algorithm 2: ComputeInitialFactors

Input: $n \times m$ Boolean matrix I and the number of initial factors *init*
Output: *init* $\times m$ Boolean matrix B

```
1  C ← m × m Boolean matrix with all entries equal to 0
2  foreach C_ij do
3  |  if I_·i ≤ I_·j and |I_·i| > 0 then
4  |  |  C_ij ← 1
5  |  end
6  end
7  remove all duplicate and empty rows from C
8  f ← 0
9  foreach row i ∈ 1, ..., m of matrix C do
10 |  if row C_i· has entry j for which C_ij = 1 and C_kj = 0 for all k < i then
11 |  |  f ← f + 1
12 |  |  add row C_i· as a new row to B
13 |  end
14 |  if f = init then
15 |  |  return B
16 |  end
17 end
```

the vectors of the association matrix serve as the candidate B-parts of factors. Now, it is easy to observe that to select the rows of the association matrix, Asso uses basically the same strategy as 8M, only more general. Where 8M tests inclusion of columns i and j (l. 3 of algorithm 2), Asso tests whether the degree of partial inclusion of column i in column j exceeds a user-specified threshold τ (or whether the confidence of the association rule $\{i\} \Rightarrow \{j\}$ exceeds τ in terms of Asso). Setting $\tau = 1$ would yield the same vectors in the association matrix of Asso as what 8M uses as the initial factors. Even though we do not explore this observation in this paper, is shall be explored further.

Algorithm 3: RefineMatricesAB

Input: Boolean matrices A, B, I, number *cost*

```
1  repeat
2  |  RefineMatrixA(A, B, I, cost)
3  |  RefineMatrixB(A, B, I, cost)
4  until loop executed 3 times or A and B did not change
```

Refining of A and B by RefineMatricesAB (algorithm 3) consists in performing a cycle until A and B do not change but at most three times, in which A is computed from I, B, and the parameter *cost* by a so-called Boolean regression described in RefineMatrixA (algorithm 4), followed by computing symmetri-

cally B using RefineMatrixB. A new factor is computed in l. 6–8 of 8M by computing first the positive part Δ^+ of the discrepancy matrix $\Delta = I - A \circ B$, one adds to A as new column the column j of Δ^+ containing the largest number of 1s, and adds to B a row of 0s with 1 at position j. For space reasons, we do not describe the meaning of Boolean regression further here; it shall be described in an extended version of this paper.

Algorithm 4: RefineMatrixA

Input: Boolean matrices A, B, I and *cost*

```
1  foreach row i ∈ {1,...,n} do
2  |  y ← I_{i_}; Z ← B; A_{i_} ← 0
3  |  repeat
4  |  |  foreach factor l ∈ 1,...,f do
5  |  |  |  m_l ← Σ_{j=1}^{m} y_j · Z_lj − cost · Σ_{j=1}^{m} (1 − y_j) · Z_lj
6  |  |  end
7  |  |  select p for which m_p = max_l m_l
8  |  |  if m_p > 0 then
9  |  |  |  A_ip ← 1
10 |  |  |  foreach j ∈ {1,...,m} do
11 |  |  |  |  if Z_pj = 1 then
12 |  |  |  |  |  Z_{_j} ← 0; y_j ← 0
13 |  |  |  |  end
14 |  |  |  end
15 |  |  end
16 |  until m_p > 0
17 end
```

3 Experimental Evaluation

3.1 Datasets and Algorithms

Our evaluation involves the real-world datasets Apj [8] (2044 × 1164, density 0.003), DNA [15] (4590 × 392, density 0.015), Emea [8] (3046 × 35, density 0.068), Chess [1] (size 3196 × 76, density 0.487), Firewall 1 [8] (365 × 709, 0.124), Firewall 2 [8] (325 × 590, 0.190), Mushroom [1] (8124 × 119, 0.193), and Paleo[4] (501 × 139, density 0.051) well known and commonly used in the literature on BMF. Note that size refers to the number of objects × number of attributes and that density is the percentage of the entries with 1 of the dataset. Moreover, we used two collections, $X1$ and $X2$, of synthetic datasets. Each collection includes 1000 randomly generated matrices obtained as Boolean products $A \circ B$ of 1000×40 and 40×500 matrices A and B which are randomly generated. The

[4] NOWpublic release 030717, available from `http://www.helsinki.fi/science/now/`.

average densities of datasets included in $X1$ is 0.15. In case of $X2$, the average densities are 0.2. An extended version shall contain more datasets but the present results are representative of the algorithms' behavior.

We used in the experimental evaluation the algorithms: TILING, ASSO, GRECONDD, HYPER, and PANDA (see section 1).

3.2 Evaluating 8M and Its Improved Version 8M+

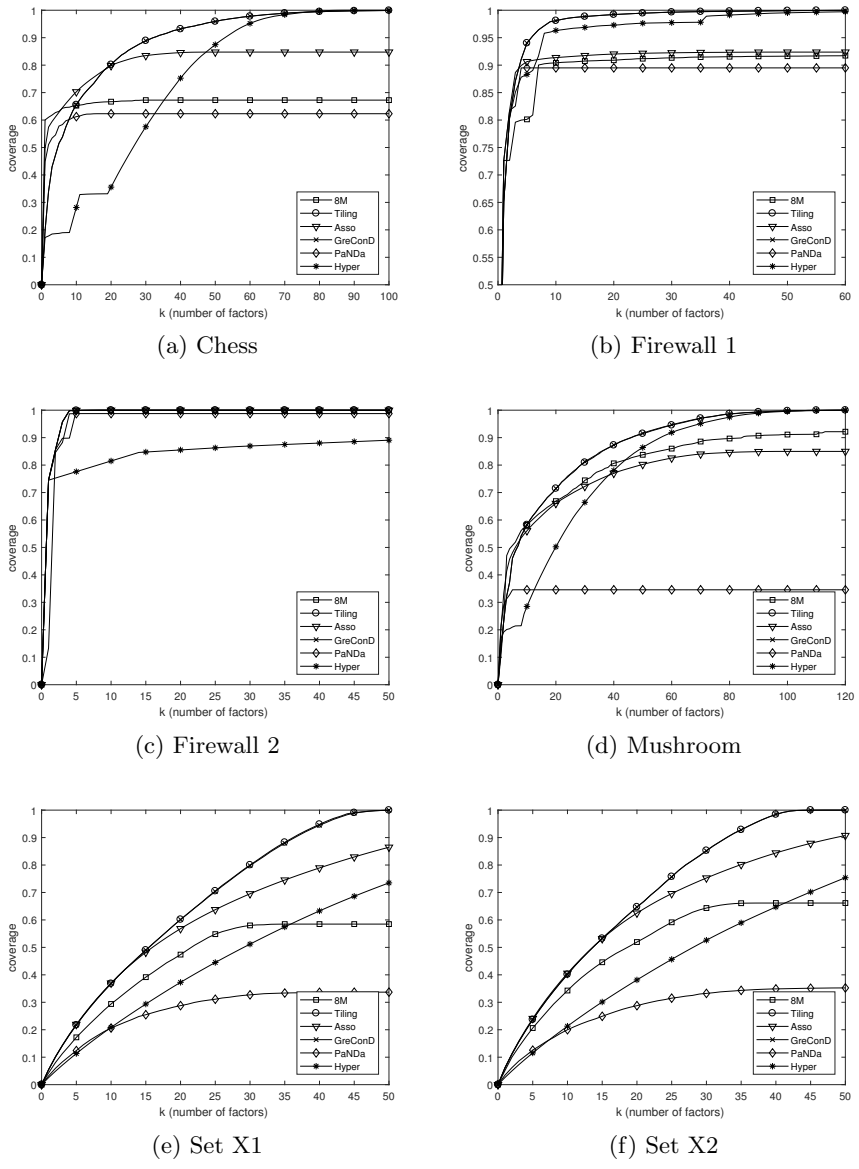
In our evaluation, we use the so-called coverage c of the input data I by the first l computed factors, i.e. the $n \times l$ and $l \times m$ matrices A and B, defined by $c(l) = 1 - E(I, A \circ B)/|I|$, in which $|I|$ is the number of 1s in I. For 8M we used the default recommendation $cost = 1$ and used various values for *init*.

Fig. 1 presents a comparison of the selected current BMF algorithms with the 8M method. The graphs depict the coverage $c(l)$ of the first l factors generated by the algorithms. One may observe that 8M compares fairly well with the current algorithms. It even outperforms PANDA on all these datasets and on most of those we experimented with. On some data, 8M outperforms ASSO and very often it outperforms HYPER in its coverage by the first few factors.

Fig. 2 presents a comparison of the basic 8M algorithm with its enhanced version denoted 8M+, which consists in a simple improvement of the initialization step of 8M. Namely, since the purpose of initialization in 8M is to obtain some reasonably good factors and since the initialization of 8M is rather simplistic, we exploited the very fast strategy of the GRECONDD algorithm to compute the first *init* factors. These have the additional advantage of committing no overcovering error. One may observe from the graphs that the improvement is significant. Moreover, taking into account Fig. 1, one can see that this improvement makes the new algorithm an interesting rival to the current algorithms.

3.3 Evaluating the Improvement of GreConD Inspired by 8M

It turns out that the idea of revisiting the previously generated factors may easily be implemented in one of the currently best BMF algorithms, GRECONDD, and yields a significant improvement as regards exact and almost exact factorizations. In our modification of GRECONDD, we revisit—every time a new factor is generated as in the original GRECONDD—the previously generated factors. If removal of a factor under consideration would result in an increase in the error E not larger than $p \times |I|$, where p is a parameter, we removed the factor. In Table 1, the columns represent the original GRECONDD and its modifications for $p = 0, 0.01, \ldots, 0.05$, the rows labeled "$k$" represent the number of factors obtained by the particular algorithm on the given dataset, and the row labeled "c" contains the coverage of the computed factorization. Thus, for instance, when factorizing the Mushroom data, the original GRECONDD needs 120 factors to obtain exact factorization. Our modification with $p = 0$ requires only 113 factors for exact factorization and only 61 factors for computing a highly accurate factorization, namely with coverage 0.951. Since such behavior is typical, we find it

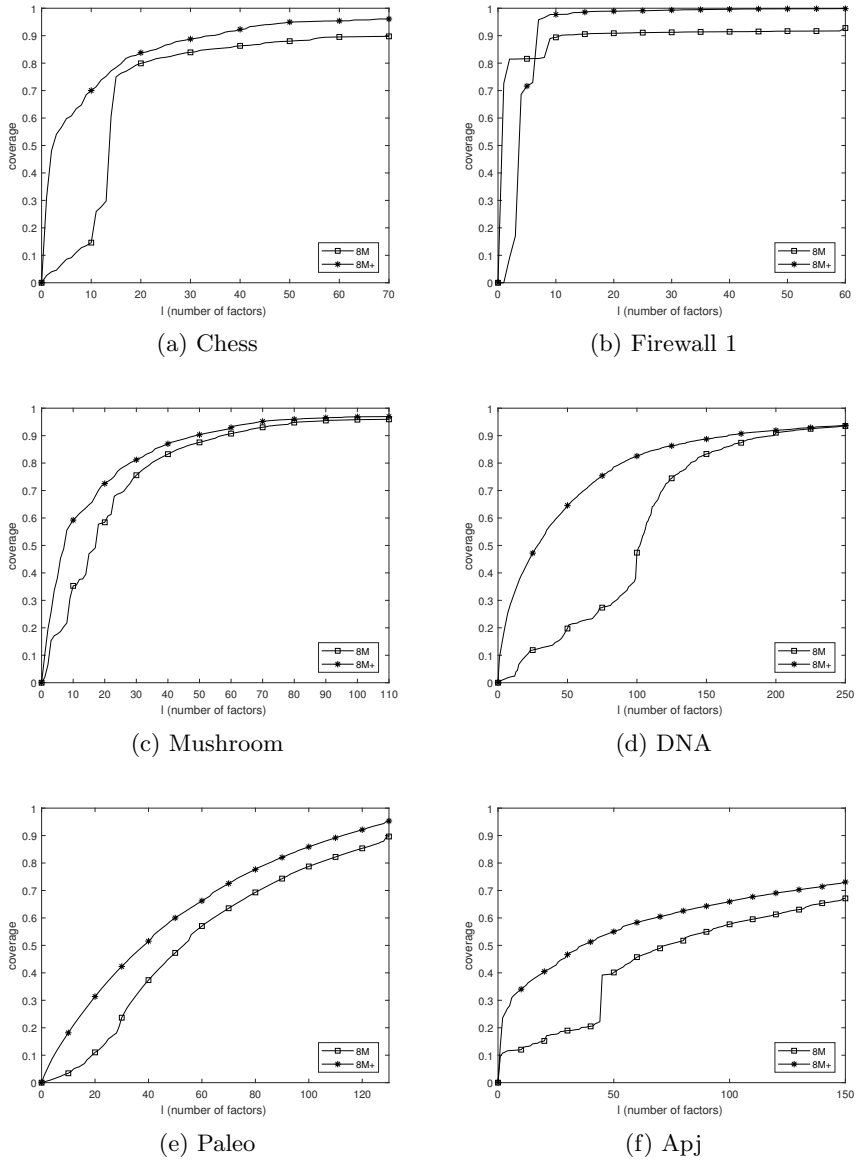


(a) Chess (b) Firewall 1

(c) Firewall 2 (d) Mushroom

(e) Set X1 (f) Set X2

Fig. 1: Coverage quality of the first l factors on real and synthetic data.



(a) Chess (b) Firewall 1

(c) Mushroom (d) DNA

(e) Paleo (f) Apj

Fig. 2: Coverage quality of the first l factors on real data: 8M vs. 8M+.

very interesting and find the idea of revisiting factors worth further exploration. Note that we did similar improvements with similar effects to Asso.

Table 1: Improvements to the GreConD algorithm

Dataset		orig.	0	0.01	0.02	0.03	0.04	0.05
Emea	k	42	34	29	26	25	24	23
	c	1.000	1.000	0.992	0.981	0.975	0.963	0.956
Chess	k	124	119	72	62	55	51	47
	c	1.000	1.000	0.991	0.981	0.970	0.962	0.952
Firewall 1	k	66	65	17	10	8	7	6
	c	1.000	1.000	0.990	0.981	0.972	0.964	0.953
Firewall 2	k	10	10	4	4	4	4	3
	c	1.000	1.000	0.998	0.998	0.998	0.998	0.958
Mushroom	k	120	113	81	73	69	65	61
	c	1.000	1.000	0.990	0.980	0.970	0.960	0.951

4 Conclusions

In addition to the fact that a description and detailed experimental evaluation of 8M were long due, we believe that the most interesting finding for future research is the property of 8M to revisit and possibly drop the previously computed factors. This idea is appealing particularly for algorithms performing general BMF, i.e. those committing overcovering error because, unlike the symmetric undercovering error, overcovering error can only increase if an algorithm does not revisit and modify the previously computed factors. Our straightforward implementation of this idea to GreConD and Asso yields an improvement which represents a promising sign of a usefulness of this idea, which hence needs to be further explored. Another topic worth further investigation is the regression procedure of 8M. While we described how it works, it is not yet properly understood why this procedure, which is analogous to statistical regression, actually works and delivers reasonable results.

Acknowledgment

R. Belohlavek acknolwedges support by the ECOP (Education for Competitiveness Operational Programme) project No. CZ.1.07/2.3.00/20.0059, which was co-financed by the European Social Fund and the state budget of the Czech Republic (the present research has been conducted in the follow-up period of this project), and by grant IGA 2018 of Palacký University Olomouc, No. IGA_PrF_2018_030.

References

1. Bache, K., Lichman, M.: UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science (2013)
2. Bartl, E., Belohlavek, R., Osicka, P., Řezanková, H.: Dimensionality Reduction in Boolean Data: Comparison of Four BMF Methods. CHDD 2012: 118–133. (2012)
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)
4. Belohlavek R., Outrata J., Trnecka M.: Impact of Boolean factorization as preprocessing methods for classification of Boolean data. Ann. Math. Artif. Intell. **72**(1-2), 3-22 (2014)
5. Belohlavek, R., Trnecka, M.: From-below approximations in Boolean matrix factorization: Geometry and new algorithm. J. Comput. Syst. Sci. **81**(8), 1678–1697 (2015)
6. Belohlavek, R., Trnecka, M.: A new algorithm for Boolean matrix factorization which admits overcovering. Discrete Applied Mathematics (in press).
7. Dixon, W. J. (ed.): BMDP Statistical Software Manual. Berkeley, CA: University of California Press (1992)
8. Ene, A., Horne, W., Milosavljevic N., Rao, P., Schreiber, R., and Tarjan R. E.: Fast exact and heuristic methods for role minimization problems. In: SACMAT 2008, pp. 1–10. (2008)
9. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: Discovery Science 2004, pp. 278–289. (2004)
10. Lu, H., Vaidya, J., Atluri, V., Hong, Y.: Constraint-aware role mining via extended Boolean matrix decomposition. IEEE Trans. Dependable and Secure Comp. **9**(5), 655–669 (2012)
11. Lucchese, C., Orlando, S., Perego, R.: Mining top-K patterns from binary datasets in presence of noise. In: SIAM DM 2010, pp. 165–176. (2010)
12. Miettinen, P.: Sparse Boolean matrix factorizations. In: IEEE ICDM 2010, pp. 935–940. (2010)
13. Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. IEEE Trans. Knowledge and Data Eng. **20**(10), 1348–1362 (2008)
14. Miettinen, P., Vreeken J.: Model order selection for Boolean matrix factorization. In: ACM SIGKDD 2011, pp. 51–59. (2011)
15. Myllykangas, S. et al: 2006, DNA copy number amplification profiling of human neoplasms. Oncogene **25**(55), 7324–7332 (2006)
16. Outrata, J.: Boolean factor analysis for data preprocessing in machine learning. In: ICMLA 2010, pp. 899–902. (2010)
17. Stockmeyer, L., The set basis problem is NP-complete, Tech. Rep. RC5431, IBM, Yorktown Heights, NY, USA (1975)
18. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: finding a minimal descriptive set of roles. In: SACMAT 2007, pp. 175–184. (2016)
19. Xiang, Y., Jin, R., Fuhry, D., Dragan, F. F.: Summarizing transactional databases with overlapped hyperrectangles. Data Mining and Knowledge Discovery **23**, 215–251 (2011)