Running Boolean Matrix Factorization in Parallel

Jan Outrata and Martin Trnecka

DEPARTMENT OF COMPUTER SCIENCE
PALACKÝ UNIVERSITY, OLOMOUC
CZECH REPUBLIC

AusDM 2016
December 6–8, Canberra

Outline

1. the problem, our contribution
2. preliminaries in Boolean Matrix Factorization (BMF)
3. running BMF in parallel
   - base (sequential) algorithm
   - **general parallelization scheme**
4. experimental evaluation
5. conclusions

Boolean Matrix Factorization (BMF) in parallel?

- BMF also called **Boolean matrix decomposition**, **Boolean factor analysis**, ...

= (approximate) decomposition of Boolean matrix (entries 1 or 0) to (Boolean) matrix product of two Boolean matrices

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

- **optimization problems**:
  1. find a decomposition with inner matrix product dimension as low as possible for a given maximal decomposition approximation = **Approximate Factorization Problem (AFP)**
  2. find as exact decomposition as possible for a given maximal inner dimension = **Discrete Basis Problem (DBP)**
- least dimension of exact decomposition = Boolean (Schein) rank of matrix
- NP-hard problems → **approximation algorithms** for sub-optimal decompositions: GreConD, GreEss (both for AFP), Asso (for DBP) and other (PaNDa, Hyper)

Boolean Matrix Factorization (BMF) in parallel?

Algorithms

- **heuristic** = final decomposition constructed from partial (approximate) decompositions which are only **locally optimal**
- **sequential** = choice of optimal partial decomposition hardcoded in algorithm design, one cannot explore several most optimal (or even all) → parallel computation (preferred – multicore CPUs, GPGPU)
- no parallel algorithm for *Boolean* matrix factorization – there are for methods designed for real-valued matrices (SVD, NMF), but they lack interpretability when applied to Boolean matrices! – crucial for knowledge discovery ⇒ BMF more appropriate for Boolean matrices
- reasons? (most commonly used) greedy heuristic approach is inherently sequential, BMF is young compared to real-valued factorization methods (?)

Our contribution

- not a parallel BMF algorithm

→ **general parallelization scheme** to compute in parallel several locally optimal decompositions and select the most optimal one(s) hoping to find the globally optimal

- following several choices of locally most optimal partial decompositions in the heuristics, constructing several most optimal final decompositions – in more processes running simultaneously in parallel
- return the single most optimal decomposition or several top-k of them
- applicable to any sequential heuristic BMF algorithm – chosen GreConD for demonstration (simple, well-known, efficient)

Boolean Matrix Factorization (BMF) – preliminaries

= (approximate) decomposition of Boolean matrix I (entries 1 or 0) to (Boolean) matrix product of two Boolean matrices A and B

$$I_{ij} \approx (A \circ B)_{ij} = \max_{l=1}^{k} \min(A_{il}, B_{lj}) \qquad \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

- I … **object-attribute incidence relation**, A … **object-factor i. r.**, B … **factor-attribute i. r.**

= discovery of k **factors** (approximately) explaining I ≈ "new attributes":
$(A \circ B)_{ij}$ … "object i has attribute j ($I_{ij} = 1$) if and only if there exists a factor l that applies to i ($A_{il} = 1$) and j is one of the manifestations of l ($B_{lj} = 1$)"

- geometric view: factor ∼ **rectangle** full of 1s → decomposition of I ∼ **coverage** of 1s of I by rectangles

Boolean Matrix Factorization (BMF) – preliminaries

- optimization problem (AFP): find a decomposition with the number k of factors as small as possible such that $||I - A \circ B|| \leq \varepsilon$ ...explain a prescribed portion of data

$$E(I, A \circ B) = ||I - A \circ B|| = \sum_{i,j=1}^{m,n} |I_{ij} - (A \circ B)_{ij}|$$

- quality of decomposition $\rightarrow$ **coverage quality** of the first l factors:

$$c(l) = 1 - E(I, A \circ B)/||I||$$

Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. *Journal of Computer and System Sciences* 76(1)(2010), 3–20.

Belohlavek R., Trnecka M.: From-Below Approximations in Boolean Matrix Factorization: Geometry and New Algorithm, *Journal of Computer and System Sciences* 81(8)(2015), 1678–1697.

GreConD – base (sequential) algorithm

Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. *Journal of Computer and System Sciences* 76(1)(2010), 3–20.

- chosen base (sequential) BMF algorithm for demonstration of our general parallelization scheme

= **greedy search** for factors – each factor explains as much of input matrix as possible, until the prescribed number of 1s is covered (i.e designed for the AFP)

- factor = **maximal rectangle** – maximal numbers of objects and attributes, stems **Formal concept analysis (FCA)** (maximal rectangle $\sim$ formal concept) $\rightarrow$ "Greedy Concepts on Demand"

GRECOND – base (sequential) algorithm

- greedy "on demand" factor/rectangle computation, not selection among candidates
  - starting empty set of attributes is repeatedly grown by a selected attribute – such that the rectangle grown by the attribute covers as many still uncovered 1s in input matrix as possible, as long as the number of 1s increases
  - other attributes may be added with the selected one – due to construction as maximal rectangle = **closure** (with all attributes shared by all objects having the attributes, see the paper)
- char. vectors of object sets of rectangles = columns of object-factor matrix A
- char. vectors of attribute sets of rectangles = rows of factor-attribute matrix B
- in details commented pseudocode in the paper (Algorithm 1)

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \quad & \quad \\ & \\ & \\ & \end{pmatrix} \circ \begin{pmatrix} \quad & \quad & \quad \\ & & \\ & & \\ & & \end{pmatrix}$$

GreConD – base (sequential) algorithm

- greedy "on demand" factor/rectangle computation, not selection among candidates
  - starting empty set of attributes is repeatedly grown by a selected attribute – such that the rectangle grown by the attribute covers as many still uncovered 1s in input matrix as possible, as long as the number of 1s increases
  - other attributes may be added with the selected one – due to construction as maximal rectangle = **closure** (with all attributes shared by all objects having the attributes, see the paper)
- char. vectors of object sets of rectangles = columns of object-factor matrix A
- char. vectors of attribute sets of rectangles = rows of factor-attribute matrix B
- in details commented pseudocode in the paper (Algorithm 1)

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \quad \end{pmatrix} \circ \begin{pmatrix} \quad \end{pmatrix}$$

GRECOND – base (sequential) algorithm

- greedy "on demand" factor/rectangle computation, not selection among candidates
  - starting empty set of attributes is repeatedly grown by a selected attribute – such that the rectangle grown by the attribute covers as many still uncovered 1s in input matrix as possible, as long as the number of 1s increases
  - other attributes may be added with the selected one – due to construction as maximal rectangle = **closure** (with all attributes shared by all objects having the attributes, see the paper)
- char. vectors of object sets of rectangles = columns of object-factor matrix A
- char. vectors of attribute sets of rectangles = rows of factor-attribute matrix B
- in details commented pseudocode in the paper (Algorithm 1)

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \quad \end{pmatrix} \circ \begin{pmatrix} \quad \end{pmatrix}$$

GreConD – base (sequential) algorithm

- greedy "on demand" factor/rectangle computation, not selection among candidates
  - starting empty set of attributes is repeatedly grown by a selected attribute – such that the rectangle grown by the attribute covers as many still uncovered 1s in input matrix as possible, as long as the number of 1s increases
  - other attributes may be added with the selected one – due to construction as maximal rectangle = **closure** (with all attributes shared by all objects having the attributes, see the paper)
- char. vectors of object sets of rectangles = columns of object-factor matrix A
- char. vectors of attribute sets of rectangles = rows of factor-attribute matrix B
- in details commented pseudocode in the paper (Algorithm 1)

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 1 & & \\ 1 & & \\ 0 & & \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ & & & & \\ & & & & \\ & & & & \end{pmatrix}$$

GreConD – base (sequential) algorithm

- greedy "on demand" factor/rectangle computation, not selection among candidates
  - starting empty set of attributes is repeatedly grown by a selected attribute – such that the rectangle grown by the attribute covers as many still uncovered 1s in input matrix as possible, as long as the number of 1s increases
  - other attributes may be added with the selected one – due to construction as maximal rectangle = **closure** (with all attributes shared by all objects having the attributes, see the paper)
- char. vectors of object sets of rectangles = columns of object-factor matrix A
- char. vectors of attribute sets of rectangles = rows of factor-attribute matrix B
- in details commented pseudocode in the paper (Algorithm 1)

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 1 & & \\ 1 & & \\ 0 & & \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ & & & & \\ & & & & \\ & & & & \end{pmatrix}$$

GRECOND – base (sequential) algorithm

- greedy "on demand" factor/rectangle computation, not selection among candidates
  - starting empty set of attributes is repeatedly grown by a selected attribute – such that the rectangle grown by the attribute covers as many still uncovered 1s in input matrix as possible, as long as the number of 1s increases
  - other attributes may be added with the selected one – due to construction as maximal rectangle = **closure** (with all attributes shared by all objects having the attributes, see the paper)
- char. vectors of object sets of rectangles = columns of object-factor matrix A
- char. vectors of attribute sets of rectangles = rows of factor-attribute matrix B
- in details commented pseudocode in the paper (Algorithm 1)

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ \boxed{1} & 1 & 0 & 1 & \boxed{1} \\ 1 & 1 & 1 & 0 & 0 \\ \boxed{1} & 0 & 0 & 0 & \boxed{1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \\ 1 & \boxed{1} & \\ 1 & 0 & \\ 0 & \boxed{1} & \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ \boxed{1} & 0 & 0 & 0 & \boxed{1} \\ & & & & \\ & & & & \end{pmatrix}$$

GRECOND – base (sequential) algorithm

- greedy "on demand" factor/rectangle computation, not selection among candidates
  - starting empty set of attributes is repeatedly grown by a selected attribute – such that the rectangle grown by the attribute covers as many still uncovered 1s in input matrix as possible, as long as the number of 1s increases
  - other attributes may be added with the selected one – due to construction as maximal rectangle = **closure** (with all attributes shared by all objects having the attributes, see the paper)
- char. vectors of object sets of rectangles = columns of object-factor matrix A
- char. vectors of attribute sets of rectangles = rows of factor-attribute matrix B
- in details commented pseudocode in the paper (Algorithm 1)

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

GreConD – base (sequential) algorithm

- greedy "on demand" factor/rectangle computation, not selection among candidates
  - starting empty set of attributes is repeatedly grown by a selected attribute – such that the rectangle grown by the attribute covers as many still uncovered 1s in input matrix as possible, as long as the number of 1s increases
  - other attributes may be added with the selected one – due to construction as maximal rectangle = **closure** (with all attributes shared by all objects having the attributes, see the paper)
- char. vectors of object sets of rectangles = columns of object-factor matrix A
- char. vectors of attribute sets of rectangles = rows of factor-attribute matrix B
- in details commented pseudocode in the paper (Algorithm 1)

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Running GreConD in parallel?

- alone factors computed by GreConD optimal (in explaining as much of input matrix as possible), but several (or all) together may be not $\Rightarrow$ partial decompositions (factor + previous factors) only locally optimal
- can be more equally optimal factors $\rightarrow$ different final decompositions – will be important in experiments later

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- alone selected attributes in factor/rectangle computation optimal (in covering as many still uncovered 1s in input matrix as possible), but several together may be not $\Rightarrow$ partial factor (attribute + previous attributes) also only locally optimal
- can also be more equally optimal attributes to select $\rightarrow$ different factors

Running a BMF algorithm in parallel

= construct, simultaneously in parallel, several (locally) most optimal partial decompositions and select among them several most optimal final decompositions, in hope to find the globally optimal one

For GRECOND

- in factor search (= decomposition construction), compute several factors explaining most of the input matrix = several locally optimal partial decompositions – **parallel computation**
- in factor computation, select several attributes so that the corresponding rectangle covers most still uncovered 1s in the input matrix = several locally optimal partial factors – **serial computation**
- in details commented pseudocode in the paper (Algorithms 2 and 3):
  - several instances of (modified) GRECOND running simultaneously in parallel processes – each (serially) computing several most optimal distinct (partial) factors

  → **GRECONDP = GRECOND in Parallel runs**
  - joint construction of several most optimal decompositions of input matrix – sorted from the most optimal one

Experimental evaluation

- comparison of GreConDP with base GreConD: quality of decomposition $\to$ **coverage quality** – most important in evaluation of performance of BMF algorithms
  1. numbers of factors for large coverage $\rightsquigarrow$ low, slowly increasing (AFP view)
  2. values of coverge for few factors $\rightsquigarrow$ high, quickly increasing to 1 (DBP view)
- comparison of GreConD with other BMF algorithms in e.g.
  - Belohlavek R., Trnecka M.: From-Below Approximations in Boolean Matrix Factorization: Geometry and New Algorithm, *Journal of Computer and System Sciences* 81(8)(2015), 1678–1697.
- in the paper examined also similarities of several (most optimal) decompositions delivered by GreConDP – they are rather similar but starting from different
- running time?: time complexity not a primary concern in BMF, GreConDP $p/2$ times slower than GreConD for p times more processes than processor units

Experimental evaluation

Datasets

1. synthetic: matrix product of randomly generated matrices with known characteristics (density, inner product dimension), enables average case evaluation
2. real: real factors, well known from various BMF papers and UCI Machine Learning Repository[1]

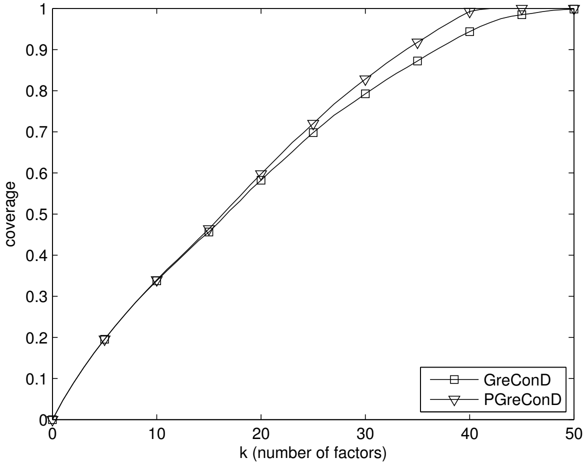
Dataset	Size	Dens. 1	Equal
Emea	3046×35	0.095	157.279
DBLP	19×6980	0.130	2.105
Firewall 1	365×709	0.124	31.168
Mushroom	8124×119	0.193	3.148
Paleo	501×139	0.051	5.868
Zoo	101×28	0.305	5.867

real datasets and their characteristics

- column **Equal** = average number of equally (locally) optimal factors per factor in GreConD – recall slide 18, **new characteristics** influencing results

[1] `archive.ics.uci.edu/ml`

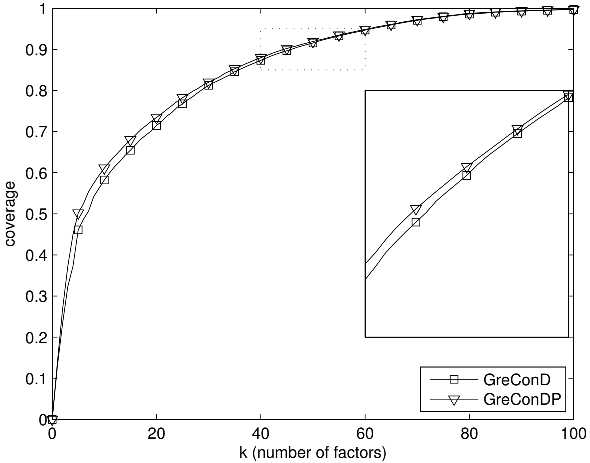
Experimental evaluation



synthetic datasets
16 parallel processes

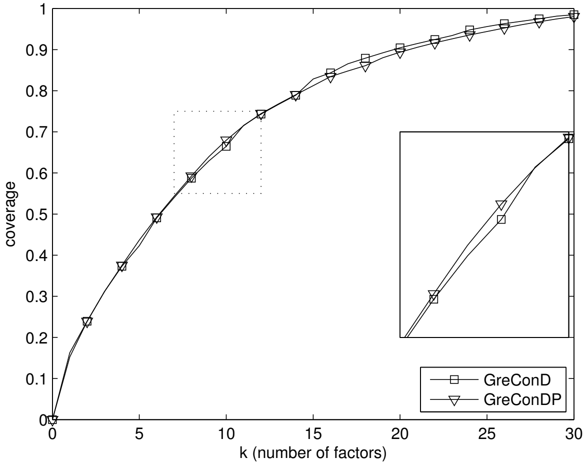
$k = 40$ . . . expected number of factors (inner matrix product dimension), delivered original factors

Experimental evaluation



Mushroom dataset
4 parallel processes

Experimental evaluation



Emea dataset,
4 parallel processes

GreConDP worse than GreConD (from AFP viewpoint) – extreme Equal characteristics, advantage of utilizing more equally optimal factors vanishes

Conclusions

- **general parallelization scheme for Boolean matrix factorization (BMF)** – applicable to any sequential heuristic BMF algorithm
- **new algorithm GreConDP** utilizing the scheme – based on simple, well-known and efficient GreConD
- in experiments GreConDP outperforms GreConD in quality of decomposition, at moderate computing time expenses
- **decomposition quality improvement** depends the number of parallel runs (higher = better) and the number of equally locally optimal factors in decomposition constructions (not much higher that the number of parallel runs)

Future research

- application to other BMF algorithms (GreEss, Asso)
- study of properties of the equally locally optimal factors – to factorize better