Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Kombinace algoritmů GreEss a Asso pro booleovskou faktorizaci matic

2020

Bc. Adam Řezníček

Vedoucí práce:
RNDr. Martin Trnečka, Ph.D.

Studijní obor: Aplikovaná informatika,
prezenční forma

Bibliografické údaje

Autor:	Bc. Adam Řezníček
Název práce:	Kombinace algoritmů GreEss a Asso pro booleovskou faktorizaci matic
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	RNDr. Martin Trnečka, Ph.D.
Počet stran:	49
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Bc. Adam Řezníček
Title:	Combination of GreEss and Asso Algorithms for Boolean Matrix Factorization
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, full-time form
Supervisor:	RNDr. Martin Trnečka, Ph.D.
Page count:	49
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Algoritmus Asso *je jedním z nejznámější algoritmů pro booleovskou faktorizaci matic; od jeho uvedení však byly představeny nové poznatky a algoritmy. Zde jsou relevantní zejména algoritmy pro aproximaci zdola, jmenovitě* GreEss *a* GreConD*, které jsou založeny na základech formální konceptuální analýzy a jsou schopny přesné dekompozice. Práce představuje nový algoritmus, pojmenovaný* Esso*, který vznikl kombinací tří zmíněných algoritmů, s nimiž je také experimentálně porovnán na reálných datech. Ačkoliv* Esso *může být považováno za modifikaci algoritmu* Asso*, protože používá jeho strukturu, liší se v použití essential části vstupní matice jako základu pro kandidátní matici a použitím greedy přístupu využívaného v algoritmu* GreConD *a* GreEss *pro vybrání nejlepšího kandidáta a konstrukci jemu odpovídajícího faktoru.*

Synopsis

The Asso *algorithm is one of the best-known algorithms for Boolean matrix factorization; however, new findings and algorithms were presented since its introduction. The most relevant for this work are the from-below approximation algorithms, namely* GreEss *and* GreConD*, which are based on the foundations of formal concept analysis, and are able to describe the input data completely. A new algorithm called* Esso*—a result of combining the three aforementioned algorithms—is presented and tested on real-world datasets. While* Esso *algorithm can be considered a modification of* Asso*, as it has the same structure, the candidate matrix is based on the essential part of the input matrix, and the process of selecting the best candidate and finding the corresponding factor is based on the greedy approach utilized in* GreConD *and* GreEss*.*

Klíčová slova: Booleovská faktorizace matic; Asso; GreConD; GreEss; Esso; Formální konceptuální analýza

Keywords: Boolean matrix factorization; Asso; GreConD; GreEss; Esso; Formal concept analysis

Děkuji RNDr. Martinu Trnečkovi, Ph.D. za umožnění práce, její vedení a za cenné rady během jejího vypracování.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

1 Úvod

Při analýze a zpracování booleovských relačních dat je přirozeným požadavkem, aby výstupem opět byla booleovská data [1] z důvodu jejich lepší interpretovatelnosti a zachování charakteru. Proto je v těchto případech vhodné používat metody faktorizace booleovských matic (BMF), oproti metodám jako je například SVD [1]. BMF algoritmy hledají skryté vztahy – *faktory* – v datech [2, 3, 4], které vystihují vstupní data, a umožňují tak jejich popis z jiného úhlu pohledu nebo snížení jejich dimenzionality [3]. Tyto faktory sestávají ze dvou složek: z množiny objektů [2, 3, 4, 5] (v [1] nazývaných pozorování) a množiny atributů [1, 2, 3, 4, 5]. Jelikož tato práce, která se snaží vylepšit algoritmus Asso [1] s využitím novějších poznatků z oblasti BMF, vychází z prací Bělohlávka, Vychodila [2], Bělohlávka, Trnečky [3] a Trnečky, Trnečkové [6], které používají formální konceptuální analýzu (FCA) [5], budou i zde používány pojmy a notace z této oblasti.

1.1 Základní pojmy z formální konceptuální analýzy

Dvouhodnotové tabulky a matice lze popsat pojmem formální kontext, což je trojice $\langle X, Y, I\rangle$, kde $X = \{x_1, \ldots, x_n\}$ a $Y = \{y_1, \ldots, y_m\}$ jsou neprázdné množiny objektů a atributů respektive a $I \subseteq X \times Y$ vyjadřuje, které objekty mají které atributy [5]. V [2, 3] se pro booleovskou matici používá značení $I \in \{0,1\}^{n\times m}$, tj. $I_{ij} = 1$ právě tehdy, když $\langle x_i, y_j\rangle \in I$. Příklad formálního kontextu ve třech různých reprezentacích je uveden na Obrázku 1.

$$X = \{x_1, x_2, x_3, x_4, x_5\}, \quad Y = \{y_1, y_2, y_3, y_4, y_5, y_6\}$$
$$I = \{\langle x_1, y_1\rangle, \langle x_1, y_4\rangle, \langle x_2, y_1\rangle, \langle x_2, y_2\rangle, \langle x_2, y_3\rangle, \langle x_2, y_5\rangle, \langle x_2, y_6\rangle, \langle x_3, y_2\rangle, \langle x_3, y_4\rangle, \langle x_3, y_6\rangle, \langle x_4, y_1\rangle, \langle x_4, y_2\rangle, \langle x_4, y_3\rangle, \langle x_4, y_4\rangle, \langle x_5, y_1\rangle, \langle x_5, y_3\rangle, \langle x_5, y_4\rangle, \langle x_5, y_6\rangle\}$$

I	y_1	y_2	y_3	y_4	y_5	y_6
x_1	×			×		
x_2	×	×	×		×	×
x_3		×		×		×
x_4	×	×	×	×		
x_5	×		×	×		×

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

Obrázek 1: Tři reprezentace formálního kontextu

Formální kontext indukuje operátory $^{\uparrow} : 2^X \to 2^Y$ a $^{\downarrow} : 2^Y \to 2^X$ [5], t.ž. pro libovolné $C \subseteq X$ a $D \subseteq Y$:

$$C^{\uparrow} = \{y \in Y \mid \forall x \in C\colon \langle x, y\rangle \in I\},$$
$$D^{\downarrow} = \{x \in X \mid \forall y \in D\colon \langle x, y\rangle \in I\}.$$

V některých případech, např. u algoritmu GreEss v části 6, je nutné specifikovat, která relace tyto operátory indukuje, k čemuž se používá značení $^{\uparrow_I}$ a $^{\downarrow_I}$ [5].

Formální koncept (v češtině někdy také „pojem“) v kontextu $\langle X, Y, I\rangle$ sestává ze dvou složek – množiny objektů nazývané *extent* a množiny atributů nazývané *intent*, tj. $\langle C, D\rangle$, kde $C \subseteq X$ a $D \subseteq Y$, pro které platí, že jsou pevným bodem operátorů $\langle^{\uparrow}, ^{\downarrow}\rangle$ [5], tj.

$$C^{\uparrow} = D \ \wedge \ D^{\downarrow} = C.$$

Množina všech formálních konceptů kontextu $\langle X, Y, I\rangle$ se značí $\mathcal{B}(X, Y, I)$ [5], která společně s uspořádáním $\leq$ definovaným pro libovolné formální koncepty $\langle C_1, D_1\rangle,\ \langle C_2, D_2\rangle \in \mathcal{B}(X, Y, I)$:

$$\langle C_1, D_1\rangle \leq \langle C_2, D_2\rangle \text{ p.k. } C_1 \subseteq C_2 (\text{nebo ekvivalentně p.k. } D_2 \subseteq D_1),$$

tvoří konceptuální svaz $\langle \mathcal{B}(X, Y, I), \leq\rangle$ [5].

Věta 1 (Hlavní věta o konceptuálních svazech, [5])

(1) $\mathcal{B}(X, Y, I)$ je úplný svaz, kde infimum a supremum jsou dány následovně:

$$\bigwedge_{j\in J} \langle A_j, B_j\rangle = \langle \bigcap_{j\in J} A_j, (\bigcup_{j\in J} B_j)^{\downarrow\uparrow}\rangle,$$
$$\bigvee_{j\in J} \langle A_j, B_j\rangle = \langle (\bigcup_{j\in J} A_j)^{\uparrow\downarrow}, \bigcap_{j\in J} B_j\rangle.$$

(2) Libovolný úplný svaz $V = \langle V, \leq\rangle$ je isomorfní s $\mathcal{B}(X, Y, I)$ p.k. existují zobrazení $\gamma : X \to V$ a $\mu : Y \to V$, t.ž.

(a) $\gamma(X)$ je supremálně hustá ve V a $\mu(V)$ je infimálně hustá ve V,

(b) $\gamma(x) \leq \mu(y)$ p.k. $\langle x, y\rangle \in I$.

Poznámka 2 (Supremální a infimální hustota, [5])
$K \subseteq V$ je supremálně hustá ve V p.k. pro každý prvek $v \in V$ existuje $K' \subseteq K$ t.ž. $v = \bigvee K'$; duálně pro infimální hustotu [5].

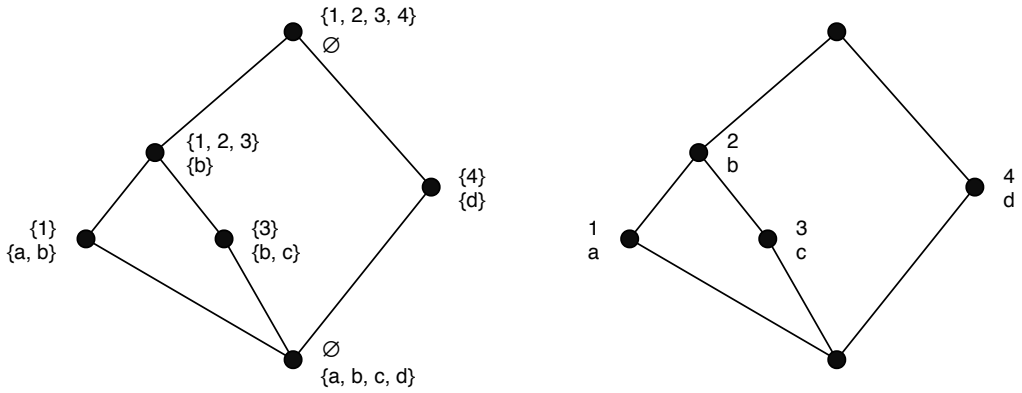
Hlavní věta o konceptuálních svazech tedy říká, že pro každou množinu konceptů $K \subseteq \mathcal{B}(X, Y, I)$ existuje supremum a infimum [2, 5]. Její druhá část dává způsob, jak lze označit jen některé z uzlů odpovídajícího *Hasseova diagramu*, aniž by došlo ke ztrátě informace, tzv. *labelled Hasseův diagram* [5] – k tomu jsou potřeba pouze *objektové* a *atributové* koncepty [5]:

$\gamma(x) = \langle \{x\}^{\uparrow\downarrow}, \{x\}^{\uparrow}\rangle$ ... objektový koncept – uzel v diagramu je označen „x“,
$\mu(y) = \langle \{y\}^{\downarrow}, \{y\}^{\downarrow\uparrow}\rangle$ ... atributový koncept – uzel v diagramu je označen „y“.

Extenty a intenty konceptu $\langle C, D \rangle$ odpovídajícímu uzlu c lze v labelled Hasseově diagramu vyčíst následovně [5] (Obrázek 2):

$x \in C$ p.k. uzel s označením x leží na cestě jdoucí od uzlu c dolů,

$y \in D$ p.k. uzel s označením y leží na cestě jdoucí od uzlu c nahoru.



(a) Hasseův diagram (b) Labelled Hasseův diagram

Obrázek 2: Hasseovy diagramy (horní množina jsou objekty, dolní atributy)

Jak ukazují následující dvě věty z [2], použití FCA, konkrétně formálních konceptů jako faktorů, je univerzální a optimální [2].

Věta 3 (Univerzálnost formálních konceptů jako faktorů, [2])
Pro každé I existuje $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$, t.ž. $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$.

Věta 4 (Optimalita formálních konceptů jako faktorů, [2])
Nechť $I = A \circ B$ pro $A \in \{0,1\}^{n \times k}$ a $B \in \{0,1\}^{k \times m}$, pak existuje $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ t.ž.

$$|\mathcal{F}| \leq k$$

a pro $A_{\mathcal{F}} \in \{0,1\}^{n \times |\mathcal{F}|}$ a $B_{\mathcal{F}} \in \{0,1\}^{|\mathcal{F}| \times m}$ platí

$$I = A_{\mathcal{F}} \circ B_{\mathcal{F}}.$$

Poznámka 5 (Množina faktorových konceptů a jí odpovídající matice, [3])
Pro množinu faktorových konceptů $\mathcal{F} = \{\langle C_1, D_1 \rangle, \ldots, \langle C_k, D_k \rangle\}$ – indexování konceptů je uvažováno jako fixní – jsou definovány matice $A_{\mathcal{F}} \in \{0,1\}^{n \times k}$ a $B_{\mathcal{F}} \in \{0,1\}^{k \times m}$ následovně:

$$(A_{\mathcal{F}})_{il} = \begin{cases} 1 & \text{pokud } i \in C_l, \\ 0 & \text{pokud } i \notin C_l, \end{cases} \quad \text{a} \quad (B_{\mathcal{F}})_{lj} = \begin{cases} 1 & \text{pokud } j \in D_l, \\ 0 & \text{pokud } j \notin D_l, \end{cases}$$

pro $l = 1, \ldots, k$ [3], to znamená, že sloupec $A_{\_l}$ je extent C_l a řádek $B_{l\_}$ je intent D_l.

1.2 Problém booleovské faktorizace matic

Obecným cílem BMF je najít pro vstupní matici $I \in \{0,1\}^{n\times m}$ matice $A \in \{0,1\}^{n\times k}$ a $B \in \{0,1\}^{k\times m}$, t.ž. $I \approx A \circ B$ [3] s počtem faktorů k co nejmenším, kde $(A \circ B)_{ij} = max_{l=1}^{k} min(A_{il}, B_{lj})$ je booleovské násobení matic [3]. Pokud navíc platí, že počet k je minimální, pro který existuje přesná dekompozice, tj. $I = A \circ B$, říká se této hodnotě *Boolean rank* nebo také *Schein rank* [1, 2, 3]. Tak jako matice I popisuje vztah objektů a atributů, a je tedy objekt-atributovou maticí, je A, popisující vztah objektů a faktorů, maticí objekt-faktorovou a B maticí faktor-atributovou [3]. Snadno srozumitelnou interpretací je následující: A popisuje, na které objekty se jaký faktor vztahuje, a B popisuje, které atributy jsou pro daný faktor charakteristické [5] (dobrý příklad tohoto uvádí [2], kde objekty jsou pacienti, atributy jsou symptomy a nalezenými faktory jsou pak nemoci nebo podezření na ně). Jelikož se jedná o aproximaci, algoritmy se dopouštějí určité chyby, která je rovna počtu pozic, ve kterých se I a $A \circ B$ neshodují, což lze vyjádřit L_1-normou následovně [3]:

$$E(C, D) = \|C - D\|_1 = \sum_{i,j=1}^{n,m} |C_{ij} - D_{ij}|.$$

Je však nutné rozlišovat dvě složky této chyby [3]. Zatímco chybu způsobenou nepokrytím pozice v I, tj. $I_{ij} = 1$ a $(A \circ B)_{ij} = 0$, lze odstranit a může se jen snižovat, chybu překrytím, tj. $I_{ij} = 0$ a $(A \circ B)_{ij} = 1$, již odstranit nelze a může se jen zvyšovat [3].

$$\begin{aligned} E(I, A \circ B) &= E_u(I, A \circ B) + E_o(I, A \circ B), \\ E_u(I, A \circ B) &= |\{\langle i, j\rangle : I_{ij} = 1,\ (A \circ B)_{ij} = 0\}|, \\ E_o(I, A \circ B) &= |\{\langle i, j\rangle : I_{ij} = 0,\ (A \circ B)_{ij} = 1\}|. \end{aligned}$$

Při návrhu algoritmu je toto odlišné chování chyb zásadní. Pokud je dovoleno zanést E_u, pak nelze obecně dosáhnout přesné dekompozice. Příkladem toho je právě Asso, které sice k těmto chybám přistupuje odlišně, ale přesto dochází k zanášení překrytí nulových pozic v I [1, 3]. Naopak u algoritmů, ve kterých dochází k výběru faktorů z množiny formálních konceptů $\mathcal{B}(X, Y, I)$, k této chybě, právě z podstaty formálních konceptů, nemůže dojít [2, 3], a jedná se tak o *aproximace zdola* [3].

Jelikož se tato práce zabývá algoritmy Asso, GreConD a GreEss, je potřeba rozlišit dvě varianty BMF problému:

DBP *Discrete Basis Problem* [1] pro parametr $k \in \mathbb{N}$ hledá takové matice $A \in \{0,1\}^{n\times k}$ a $B \in \{0,1\}^{k\times m}$, které minimalizují $E(I, A \circ B)$. Jedná se tedy o hledání prvních k významných faktorů [1, 3].

AFP *Approximate Factorization Problem* [2, 3] pro parametr $\epsilon \geq 0$ hledá matice $A \in \{0,1\}^{n\times k}$ a $B \in \{0,1\}^{k\times m}$ s co nejmenším k, t.ž. $E(I, A \circ B) \leq \epsilon$. Důraz je tedy kladen na popsání určité části původních dat [3].

Motivací této práce jsou experimentální výsledky z [6], kde je představena metoda redukce vstupních dat s využitím *essential elements* [3], které jsou zde popsány v části 6, po jejíž aplikaci algoritmus Asso stále dosahoval dobrých výsledků. V jednom případě – na datasetu Mushroom s 50% redukcí – dokonce dosáhl lepší faktorizace oproti neredukovaným datům [6].

1.3 Algoritmy pro booleovskou faktorizaci matic

1.3.1 Asso

Algoritmus Asso byl představen jako jednoduchý postup pro řešení DBP problému [1], který využívá korelací mezi sloupci vstupní matice [1]. Mimo počet faktorů $k \in \mathbb{N}$ je parametrizován také hodnotou $\tau \in [0, 1]$ a vahami $w^+, w^- \in \mathbb{R}$, které jsou použity při konstrukci asociační matice [1], která je zde přeznačena na C[1]. Jelikož vstupní matice I má rozměry $n \times m$, pak C, která vznikne výpočtem korelací mezi sloupci I, má rozměry $m \times m$, a platí $C_{ij} = 1$ p.k. *confidence* asociačního pravidla $i \to j$ nabývá hodnoty alespoň τ, což je v [1] označováno jako „τ-*strong*“, tedy že asociace mezi sloupcem i a j matice I je τ-silná, tj.

$$c(i \to j, I) = \frac{I_{\_i} \cdot I_{\_j}}{I_{\_i} \cdot I_{\_i}} \geq \tau.$$

Řádky matice C jsou kandidáty[2], z nichž jeden je v každé iteraci vybrán jako řádek matice B [1], a popisuje tedy faktor-atributový vztah. Výběr je prováděn *greedy* přístupem na základě *cover* funkce využívající parametry w^+ a w^- s následujícím předpisem [1, 7]:

$$w^+|\{\langle i, j\rangle : I_{ij} = 1,\ (A \circ B)_{ij} = 1\}| - w^-|\{\langle i, j\rangle : I_{ij} = 0,\ (A \circ B)_{ij} = 1\}|,$$

ale jak uvádí Trnečka [7], tento popis je neúplný, neboť zde není zohledněna již pokrytá část matice a zejména není popsáno, jak se k řádku matice B hledá sloupec matice A [7].

Implementace dle [7] počítá pro každého kandidáta c *cover* funkci na každém řádku $I_{i\_}$ zvlášť [7], odděleně se sečtou pozice, které kandidát c správně pokrývá, a ty, které naopak špatně překrývá, a vynásobí se příslušnými vahami, w^+ a w^- resp. Je-li však už nějaká $\langle i, j\rangle$ pokryta (ať už správně nebo ne), to znamená $(A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij} = 1$, pak se v těchto součtech neobjeví, resp. její hodnota je vždy rovna 0. Výslednou hodnotou *cover* kandidáta c na příslušném řádku $I_{i\_}$ je pak jejich rozdíl (od správného pokrytí se odečítá chybné překrytí) [7]. Pro celkový *cover* kandidáta c je potřeba uvážit, jakým způsobem pak vzniká odpovídající sloupec a (Algoritmus 2) v matici A – pro něj platí, že $a_i = 1$, pokud je výsledek *cover* pro řádek $I_{i\_}$ větší než 0 [7]. Jelikož tedy dochází k pokrývání pozic pouze

[1] Důvodem pro tuto změnu je konflikt ve značení asociační matice v [1] se zde používaným značením, přejatým z [2, 3], pro objekt-faktorovou matici.

[2] Odtud zde používané značení C (“candidate”) namísto A (“association”) z [1].

na řádcích, kde byl výsledek *cover* kladný, celkový výsledek *cover* pro kandidáta c je roven součtu hodnot pouze těchto řádků [7].

Předpis *cover* funkce pro složku správně pokrytých pozic by tedy po doplnění a úpravě na zde používanou notaci mohl vypadat následovně:

$$w^+|\{\langle i,j\rangle : \underbrace{I_{ij}=1}_{\text{context}}, \quad \underbrace{(A\circ B)_{ij}=1}_{\text{not covered}} \quad \underbrace{}_{\text{candidate}}\}|$$

$$w^+|\{\langle i,j\rangle : \overbrace{I_{ij}=1}, \overbrace{(A_{\mathcal{F}}\circ B_{\mathcal{F}})_{ij}=0}, \overbrace{c_j=1} \quad \}|,$$

duálně pro překryté pozice (změní se w^+ na w^- a $I_{ij}=1$ na $I_{ij}=0$). Jelikož je zde v pseudokódu (Algoritmus 1) reflektována změna týkající se výpočtu *cover* funkce pro každý řádek zvlášť a výsledkem Cover není skalár, ale vektor hodnot *cover* pro jednotlivé řádky, je na řádku 8 potřeba provést součet jeho kladných složek, které jsou zde značeny jako $\text{Cover}_{>0}$.

Algoritmus 1: Cover

Input: object-factor matrix $A_{\mathcal{F}}$, factor-attribute matrix $B_{\mathcal{F}}$, $I \in \{0,1\}^{n\times m}$, candidate vector c, $w^+, w^- \in \mathbb{R}$
Output: $cover \in \mathbb{R}^n$

1 **for** $i \leftarrow 1,\ldots,n$ **do**
2 $\quad covered \leftarrow |\{\langle i,j\rangle : I_{ij}=1,\ (A_{\mathcal{F}}\circ B_{\mathcal{F}})_{ij}=0,\ c_j=1\}|$
3 $\quad overcovered \leftarrow |\{\langle i,j\rangle : I_{ij}=0,\ (A_{\mathcal{F}}\circ B_{\mathcal{F}})_{ij}=0,\ c_j=1\}|$
4 $\quad cover_i \leftarrow w^+ \cdot covered - w^- \cdot overcovered$
5 **end**
6 **return** *cover*

Algoritmus 2: ComputeObjectFactor

Input: object-factor matrix $A_{\mathcal{F}}$, factor-attribute matrix $B_{\mathcal{F}}$, $I \in \{0,1\}^{n\times m}$, candidate c, $w^+, w^- \in \mathbb{R}$
Output: $a \in \{0,1\}^{n\times 1}$

1 $cover \leftarrow \text{Cover}(A_{\mathcal{F}}, B_{\mathcal{F}}, I, c, w^+, w^-)$
2 **for** $i \leftarrow 1,\ldots,n$ **do**
3 $\quad a_i \leftarrow 1$ **if** $cover_i = 1$ **else** 0
4 **end**
5 **return** a

Jak již bylo zmíněno v sekci 1.2, algoritmus Asso sice přistupuje k chybám E_o a E_u zvlášť v rámci funkce *cover* nastavitelnými vahami w^+ a w^-, ale samotnému zanesení chyby překrytím nebrání, což znamená, že tento algoritmus obecně neposkytuje přesnou dekompozici. Tato vlastnost bude později adresována.

Příklad 6

Následující představují pro $w^+ = w^- = 1$, kandidáta $c = (0\,1\,1\,1\,0\,0)$ a I z Obrázku 1 výpočet *cover*, přičemž v prvním případě není dosud pokryta žádná pozice v příslušné submatici I_c kandidáta c a ve druhém jsou již některé pozice pokryty (vyznačeny –).

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

0 1 1 1 0 0	
0 0 1	−1
1 1 0	1
1 0 1	1
1 1 1	3
0 1 1	1
	= 6

0 1 1 1 0 0	
0 0 1	−1
– – 0	−1
1 0 1	1
– – 1	1
0 1 1	1
	= 3

Algoritmus 3: Asso

Input: Boolean matrix $I \in \{0,1\}^{n \times m}$, $k \in \mathbb{N}$, $\tau \in [0,1]$, $w^-, w^+ \in \mathbb{R}$
Output: Factor set $\mathcal{F}$

for $i \leftarrow 1, \ldots, n$ **do**
  for $j \leftarrow 1, \ldots, m$ **do**
    $C_{ij} \leftarrow 1$ **if** $c(i \to j, I) \geq \tau$ **else** 0
  end
end
$\mathcal{F} \leftarrow \emptyset$
for $l \leftarrow 1, \ldots, k$ **do**
  $b \leftarrow arg\,max_{C_{i\_} \in \{0,1\}^{1 \times m}} \sum \text{Cover}_{>0}(A_{\mathcal{F}}, B_{\mathcal{F}}, I, C_{i\_}, w^+, w^-)$
  $a \leftarrow \text{ComputeObjectFactor}(A_{\mathcal{F}}, B_{\mathcal{F}}, I, b, w^+, w^-)$
  add $\langle a, b \rangle$ to $\mathcal{F}$
end
return $\mathcal{F}$

Pokud se odhlédne od detailů algoritmu Asso[3] (Algoritmus 3), tak to, co se v něm děje, je výběr submatice na základě kandidátního vektoru, tj. vyberou se sloupce matice I, na kterých má kandidát hodnotu 1, na tuto submatici se při zohlednění již pokryté části aplikuje funkce pro výpočet skóre a pokud je maximální, tak se na základě této submatice zkonstruuje faktor. Tento pohled je uplatněn později v sekci 3.1. Pro výběr submatice matice I na základě vektoru c bude používáno značení I_c.

[3] Miettinen et al. [1] uvádí i další varianty algoritmu Asso, jmenovitě: Asso+trans, Asso+clust, Asso+opt a Asso+iter, které však v této práci nejsou uvažovány.

1.3.2 GreConD

Dvě stěžejní věty algoritmu GreConD – formální koncepty jsou univerzální a optimální faktory – byly již uvedeny v části 1.1. Dále je důležitý geometrický pohled na problém, tj. že formální koncepty odpovídají intuitivnímu pojmu maximálního „obdélníku“ (vzhledem k $\sqsubseteq$, viz Definice 7 a Věta 8), který tvoří 1 v kontextu $\langle X, Y, I \rangle$ při použití správné permutace řádků a sloupců [2, 3, 5] (Obrázek 3).

Definice 7 (Obdélník v $\langle X, Y, I \rangle$, [5])
Obdélník ve formálním kontextu $\langle X, Y, I \rangle$ je dvojice $\langle C, D \rangle$ t.ž. $C \times D \subseteq I$. Pro obdélníky $\langle C_1, D_1 \rangle$ a $\langle C_2, D_2 \rangle$ platí $\langle C_1, D_1 \rangle \sqsubseteq \langle C_2, D_2 \rangle$ p.k. $C_1 \subseteq C_2$ a $D_1 \subseteq D_2$.

I	y_1	y_2	y_3	y_4	y_5
x_1	×	×	×	×	×
x_2			×	×	
x_3		×	×	×	×
x_4			×	×	×
x_5	×			×	

Obrázek 3: Příklad dvou maximálních obdélníků

Věta 8 (Formální koncept je maximální obdélník, [5])
$\langle C, D \rangle \in \mathcal{B}(X, Y, I)$ p.k. $\langle C, D \rangle$ je maximálním obdélníkem vzhledem k $\sqsubseteq$.

V terminologii matic lze $J \in \{0,1\}^{n \times m}$ označit za obdélníkovou, pokud ji lze vyjádřit jako produkt sloupcového a řádkového vektoru, tj. $J = C \circ D$ pro $C \in \{0,1\}^{n \times 1}$ a $D \in \{0,1\}^{1 \times m}$ respektive [3]. Uspořádání pro obdélníkové matice, které odpovídá uspořádání $\langle C_1, D_1 \rangle \sqsubseteq \langle C_2, D_2 \rangle$, je následovné [3]:

$$J_1 \leq J_2 \ (J_1 \text{ je obsažená v } J_2) \quad \text{p.k.} \quad (J_1)_{ij} \leq (J_2)_{ij} \text{ pro každé } i, j.$$

Přirozeně volba obdélníků, které jsou maximální, vede k minimalizaci jejich počtu k (viz Věta 4 a 8) [2, 5]. Jak uvádí [3], následující jsou ekvivalentní:

(a) $I = A \circ B$ pro nějaké $A \in \{0,1\}^{n \times k}$ a $B \in \{0,1\}^{k \times m}$.

(b) Existují obdélníky $J_1, \ldots, J_k \in \{0,1\}^{n \times m}$ t.ž. $I = J_1 \vee \ldots \vee J_k$, tedy $I_{ij} = max_{l=1}^{k}(J_l)_{ij}$.

(c) Existují obdélníky $J_1, \ldots, J_k \in \{0,1\}^{n \times m}$ obsažené v I t.ž. $I_{ij} = 1$ p.k. pozice $\langle i, j \rangle$ je pokryta některým z obdélníků J_l.

Tento pohled umožňuje redukci BMF problému na *set cover*, jak je ukázáno v [2], pro který existuje aproximační *greedy* algoritmus s polynomiální časovou složitostí, který poskytuje téměř optimální výsledky [2]. Právě postup této aproximace je využit v algoritmu GreCon (pseudokód zde není uveden) [2], ve kterém je v každé iteraci vybrán ten formální koncept, který maximalizuje pokrytí, což znamená, že je potřeba předem spočítat $\mathcal{B}(X, Y, I)$ [2]. Bělohlávek, Vychodil [2] také uvádí nutnost zahrnout průnik množiny všech objektových a atributových konceptů do množiny $\mathcal{F}$, má-li být $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$, tedy

$$\begin{aligned}
\mathcal{O}(X, Y, I) &= \{\gamma(x) : x \in X\}, \\
\mathcal{A}(X, Y, I) &= \{\mu(y) : y \in Y\}, \\
\mathcal{O}(X, Y, I) &\cap \mathcal{A}(X, Y, I) \subseteq \mathcal{F}.
\end{aligned}$$

Tento krok sice není nutné provádět explicitně, jak uvádí [2], protože by stejně došlo k jejich zahrnutí, ale dojde tak ke zrychlení GreCon [2, 7], zařadí-li se tento krok před samotné *greedy* hledání faktorů [2, 7].

Výpočet $\mathcal{B}(X, Y, I)$ je však náročný i při použití rychlých algoritmů, nicméně tomuto výpočtu se lze vyhnout, aniž by došlo k výrazné ztrátě kvality aproximace [2]. GreConD, tedy GreCon „on demand“, pracuje tak, že dokud není pokryta celá matice I, hledá faktory tím, že k zpočátku prázdnému intentu postupně přidává „slibné“ sloupce *greedy* přístupem. Dochází tedy k přidání atributu do intentu a aplikování šipkových operátorů, přičemž vybrán je ten atribut, který maximalizuje:

$$((D \cup \{y\})^{\downarrow} \times (D \cup \{y\})^{\downarrow\uparrow}) \cap \mathcal{U},$$

kde $\mathcal{U}$ je nepokrytá část vstupní matice I. Jakmile nelze přidat další atribut, je faktor přidán do $\mathcal{F}$ a začíná další iterace. Tímto jednoduchým postupným přidáváním „slibných“ sloupců je možné zkonstruovat jakýkoliv formální koncept $\langle C, D \rangle$ kontextu daného maticí I, protože každý koncept $\langle C, D \rangle$ je možné vyjádřit jako $C = D^{\downarrow}$ a $D = \bigcup_{y \in D} \{y\}^{\downarrow\uparrow}$ [2, 3].

Zde uvedený pseudokód (Algoritmus 4) hledá přesnou dekompozici, jednoduchou modifikací je však možné získat GreConD pro DBP nebo AFP, zastavením v případě, že $|\mathcal{F}| = k$, a nebo když $E(I, A_{\mathcal{F}} \circ B_{\mathcal{F}}) < \epsilon$, respektive.

Algoritmus 4: GreConD

Input: Boolean matrix $I \in \{0,1\}^{n \times m}$
Output: Set $\mathcal{F}$ of factors

$\mathcal{U} \leftarrow \{\langle i, j\rangle | I_{ij} = 1\}$
$\mathcal{F} \leftarrow \emptyset$
while $\mathcal{U} \neq \emptyset$ **do**
  $D \leftarrow \emptyset$
  $V \leftarrow 0$
  while $\exists j \notin D$ *s.t.* $|((D \cup \{j\})^{\downarrow} \times (D \cup \{j\})^{\downarrow\uparrow}) \cap \mathcal{U}| > V$ **do**
    $D \leftarrow (D \cup \{j\})^{\downarrow\uparrow}$
    $V \leftarrow |(D^{\downarrow} \times D) \cap \mathcal{U}|$
  end
  $C \leftarrow D^{\downarrow}$
  add $\langle C, D\rangle$ **to** $\mathcal{F}$
  foreach $\langle i, j\rangle \in C \times D$ **do**
    remove $\langle i, j\rangle$ **from** $\mathcal{U}$
  end
end
return $\mathcal{F}$

1.3.3 GreEss

Zásadním rozdílem algoritmu GreEss od předchozích je, že rozlišuje důležitost pozic, a zaměřuje se tak jen na určitou podmnožinu $\mathcal{B}(X, Y, I)$, jejíž pokrytí garantuje pokrytí celé matice I [3], popř. její části, je-li $\epsilon > 0$ (zde je ale uvažována volba $\epsilon = 0$). Prvním krokem je výpočet *essential* části matice I, která je značena $\mathcal{E}(I)$ [3] a platí pro ni [3]:

$$\mathcal{E}(I)_{ij} = 1 \text{ p.k. } \mathcal{I}_{ij} \text{ je neprázdný a minimální interval vzhledem k } \subseteq .$$

Intervalem v konceptuálním svazu $\langle \mathcal{B}(X, Y, I), \leq \rangle$ se myslí podmnožina $\mathcal{B}(X, Y, I)$, která je vymezena dvojicí konceptů $\langle C_1, D_1\rangle, \langle C_2, D_2\rangle \in \mathcal{B}(X, Y, I)$:

$$[\langle C_1, D_1\rangle, \langle C_2, D_2\rangle] = \{\langle E, F\rangle \in \mathcal{B}(X, Y, I) \mid \langle C_1, D_1\rangle \leq \langle E, F\rangle \leq \langle C_2, D_2\rangle\},$$

z nichž zvlášť důležité jsou ty vymezené objektovým a atributovým konceptem $\gamma(i)$ a $\mu(j)$ resp., pro něž bylo zavedeno značení [3]:

$$\mathcal{I}_{ij} = [\gamma(i), \mu(j)].$$

Pro intervaly $\mathcal{I}_{ij}$ a $\mathcal{I}_{i'j'}$ platí [3]:

$$\mathcal{I}_{ij} \subseteq \mathcal{I}_{i'j'} \text{ p.k. } \gamma(i') \leq \gamma(i) \text{ a } \mu(j) \leq \mu(j') \text{ p.k. } \{i\}^{\uparrow} \subseteq \{i'\}^{\uparrow} \text{ a } \{j\}^{\downarrow} \subseteq \{j'\}^{\downarrow}.$$

Důležité vlastnosti intervalů následují [3]:

Lemma 9 (Vlastnosti intervalů, [3])

(a) $\mathcal{I}_{C,D}$ *je neprázdný p.k.* $C \times D \subseteq I$*, tj. pokud* $I_{ij} = 1$ *pro všechna* $i \in C$ *a* $j \in D$*. Zejména* $\mathcal{I}_{ij}$ *je neprázdný p.k.* $I_{ij} = 1$*.*

(b) $\mathcal{I}_{C,D} = \{\langle E, F\rangle \in \mathcal{B}(X, Y, I) \mid C \subseteq E, D \subseteq F\} = \{\langle E, F\rangle \in \mathcal{B}(X, Y, I) \mid C^{\uparrow\downarrow} \subseteq E, D^{\downarrow\uparrow} \subseteq F\}$*. Zejména* $\mathcal{I}_{ij}$ *je množina všech konceptů, které pokrývají pozici* $\langle i, j\rangle$*.*

(c) Pokud $(A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij} = 1$*, pak* $\mathcal{F}$ *obsahuje alespoň jeden koncept z intervalu* $\mathcal{I}_{ij}$*.*

K tomu, aby byl pokryt celý kontext $\langle X, Y, I\rangle$, resp. jemu odpovídající matice I, stačí vybrat z každého neprázdného intervalu jeden formální koncept do $\mathcal{F}$ [3], a jelikož je požadován minimální počet faktorů (1.2), má smysl se soustředit pouze na intervaly, které jsou neprázdné a *minimální* vzhledem k $\subseteq$ (Obrázek 4). Jak bylo uvedeno u algoritmu GreConD (Algoritmus 4), některé koncepty jsou „povinné“, tj. pro přesnou dekompozici $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$ je nutné je zahrnout do $\mathcal{F}$ [2], což se v $\mathcal{B}(X, Y, I)$ projevuje v podobě intervalů s $|\mathcal{I}_{ij}| = 1$, což jsou právě ty, které splňují $\gamma(i) = \mu(j)$.

Lemma 10 uvádí způsob výpočtu $\mathcal{E}(I)$ [3].

Lemma 10 (Výpočet $\mathcal{E}(I)$, [3])

$\mathcal{E}(I)_{ij} = 1$ *právě tehdy, když jsou splněny následující podmínky:*

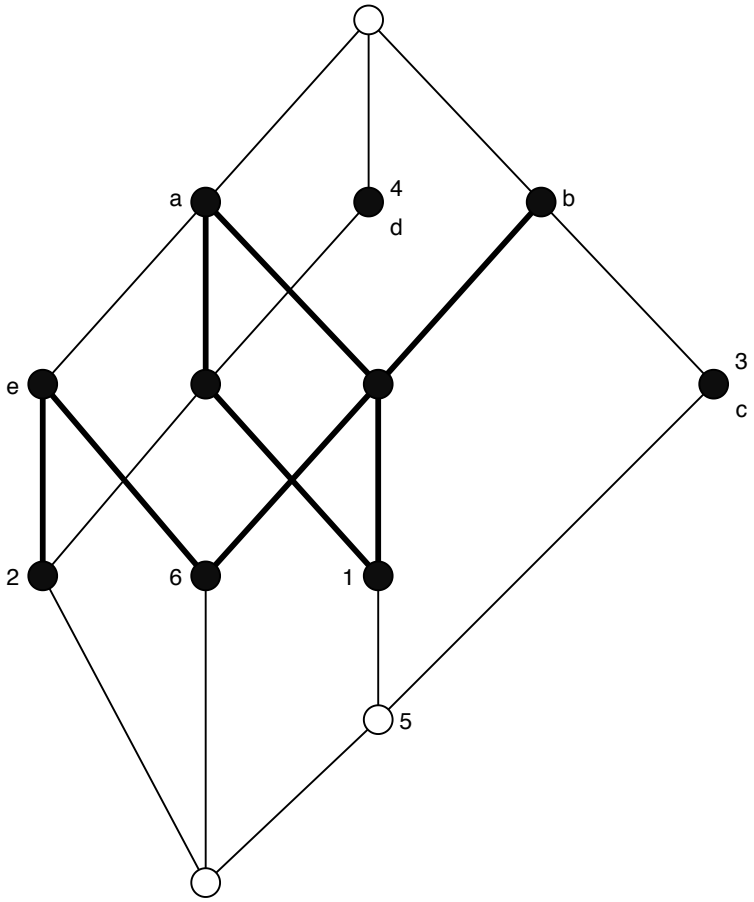
(a) $I_{ij} = 1$*,*

(b) $\forall i'$ *: pokud* $\{i'\}^{\uparrow} \subset \{i\}^{\uparrow}$ *pak* $I_{i'j} = 0$*, tj. řádek odpovídající objektu* i' *neobsahuje atribut* j*,*

(c) $\forall j'$ *: pokud* $\{j'\}^{\downarrow} \subset \{j\}^{\downarrow}$ *pak* $I_{ij'} = 0$*, tj. sloupec odpovídající atributu* j' *neobsahuje objekt* i*.*

Podmínka (a) Lemmatu 10 tedy vyjadřuje, že interval $\mathcal{I}_{ij}$ je neprázdný, podmínky (b) a (c) pak, že tento interval je minimální, tj. že neexistují i' a j' t.ž. by $\mathcal{I}_{i'j'}$ byl obsažený v $\mathcal{I}_{ij}$. *Essential* část matice I tedy říká, na jakou část konceptuálního svazu se soustředit při hledání faktorů [3], konkrétně se jedná o $\mathcal{B}_{\mathcal{E}}(I) = \bigcup\{\mathcal{I}_{ij} \mid \mathcal{E}(I)_{ij} = 1\}$ [3], což je obsahem Obrázku 4.

$$\begin{pmatrix} \underline{1} & \underline{1} & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & \underline{1} \\ 0 & 1 & \underline{1} & 0 & 0 \\ 0 & 0 & 0 & \underline{1} & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & \underline{1} & 0 & 0 & 1 \end{pmatrix}$$



Obrázek 4: Matice I a jí daný konceptuální svaz s vyznačenou *essential* částí [3]

Algoritmus GreEss využívá faktu, že $\|\mathcal{E}(I)\| \leq \|I\|$ [3] – většinou je však tento počet výrazně nižší [3], jak také ukazuje Tabulka 1 v části 2 – pro nalezení přesné faktorizace $\mathcal{G} \subseteq \mathcal{B}(\mathcal{E}(I))$, tj. $\mathcal{E}(I) = A_\mathcal{G} \circ B_\mathcal{G}$, z níž následně vypočítá faktorizaci $\mathcal{F}$ původní matice [3], jak popisuje následující věta z [3]:

Věta 11 (Výpočet faktorizace I z faktorizace $\mathcal{E}(I)$, [3])
Nechť $\mathcal{G} \subseteq \mathcal{B}(\mathcal{E}(I))$ je množina faktorových konceptů matice $\mathcal{E}(I)$, tj. $\mathcal{E}(I) = A_\mathcal{G} \circ B_\mathcal{G}$. Pak každá množina $\mathcal{F} \subseteq \mathcal{B}(I)$, která pro každý koncept $\langle C, D\rangle \in \mathcal{G}$ obsahuje alespoň jeden koncept z intervalu $\mathcal{I}_{C,D}$ je množinou faktorových konceptů matice I, tj. $I = A_\mathcal{F} \circ B_\mathcal{F}$.

Výpočet $\mathcal{G} \subseteq \mathcal{B}(\mathcal{E}(I))$ je prováděn procedurou ComputeIntervals [3] (Algoritmus 5). Zajímavé je, že množina $\mathcal{G}$ nemusí být přesnou faktorizací $\mathcal{E}(I)$, tj. $A_\mathcal{G} \circ B_\mathcal{G} < \mathcal{E}(I)$, a přesto z ní lze získat přesnou faktorizaci $\mathcal{F}$ matice I, tj. $A_\mathcal{F} \circ B_\mathcal{F} = I$ [3]. I v takovém případě totiž lze z $\mathcal{G}$ získat množinu $\mathcal{F} = \{c_{\langle C,D\rangle} \mid \langle C, D\rangle \in \mathcal{G}\}$, kde $c_{\langle C,D\rangle}$ je libovolný koncept vybraný z intervalu $\mathcal{I}_{C,D}$, pro kterou platí $I = A_\mathcal{F} \circ B_\mathcal{F}$ [3]. V proceduře ComputeIntervals, která používá postup algoritmu GreConD, se to projevuje tak, že jsou uvažovány

operátory indukované *essential* částí $\mathcal{E}(I)$ i původním kontextem I, tj. při pokrývání $\mathcal{U}$ se uvažuje $((D\cup\{j\})^{\downarrow\varepsilon})^{\uparrow_I\downarrow_I}\times((D\cup\{j\})^{\downarrow\varepsilon\uparrow\varepsilon})^{\downarrow_I\uparrow_I}$ [3], tedy navíc přibyly aplikace uzávěrů $^{\uparrow_I\downarrow_I}$ a $^{\downarrow_I\uparrow_I}$.

Algoritmus 5: ComputeIntervals

Input: Boolean matrix $I \in \{0,1\}^{n\times m}$
Output: Set $\mathcal{G} \subseteq \mathcal{B}(\mathcal{E}(I))$

1 $\mathcal{E} \leftarrow \mathcal{E}(I)$
2 $\mathcal{U} \leftarrow \{\langle i,j\rangle \mid \mathcal{E}_{ij}=1\}$
3 $\mathcal{G} \leftarrow \emptyset$
4 **while** $\mathcal{U} \neq \emptyset$ **do**
5 $\quad D \leftarrow \emptyset$
6 $\quad s \leftarrow 0$
7 $\quad$ **while** $\exists j \notin D$ *s.t.* $|((D\cup\{j\})^{\downarrow\varepsilon})^{\uparrow_I\downarrow_I}\times((D\cup\{j\})^{\downarrow\varepsilon\uparrow\varepsilon})^{\downarrow_I\uparrow_I}\cap\mathcal{U}| > s$ **do**
8 $\quad\quad$ select j maximizing $|((D\cup\{j\})^{\downarrow\varepsilon})^{\uparrow_I\downarrow_I}\times((D\cup\{j\})^{\downarrow\varepsilon\uparrow\varepsilon})^{\downarrow_I\uparrow_I}\cap\mathcal{U}|$
9 $\quad\quad C \leftarrow (D\cup\{j\})^{\downarrow\varepsilon}$
10 $\quad\quad D \leftarrow (D\cup\{j\})^{\downarrow\varepsilon\uparrow\varepsilon}$
11 $\quad\quad s \leftarrow |C^{\uparrow_I\downarrow_I}\times D^{\downarrow_I\uparrow_I}\cap\mathcal{U}|$
12 $\quad$ **end**
13 $\quad$ **add** $\langle C,D\rangle$ **to** $\mathcal{G}$
14 $\quad \mathcal{U} \leftarrow \mathcal{U} - C^{\uparrow_I\downarrow_I}\times D^{\downarrow_I\uparrow_I}$
15 **end**
16 **return** $\mathcal{G}$

GreEss následně, dokud nepokryje celou matici (nebo její část, je-li $\epsilon > 0$), provádí *greedy* výběr, inspirovaný algoritmem GreConD [3], při kterém je z každého intervalu vybrán nejvýše jeden koncept [3]. V každé iteraci se tak pro každý koncept $\langle C,D\rangle \in \mathcal{G}$ přidává k atributovému konceptu $\mu(j) \in \mathcal{I}_{C,D}$ další atributy (řádky 10-15, Algoritmus 6), které maximalizují pokrytí $\mathcal{U}$, což spěje k nalezení konceptu $\langle E,F\rangle$, který, pokud poskytuje lepší pokrytí $s_{\langle C,D\rangle}$, než je dosud nalezené maximum s, se stává aktuálním kandidátem na další faktor (řádky 16-20) [3]. Důležité je, že během tohoto procesu se používají operátory indukované pouze částí matice I, která odpovídá danému intervalu, tj. $I\cap(D^{\downarrow_I}\times C^{\uparrow_I})$ [3] (viz Lemma 12 [3]). Po skončení tohoto cyklu (řádky 6-21) je nejlepší nalezený koncept přidán do množiny $\mathcal{F}$ (řádek 22) a příslušný interval je odstraněn z množiny $\mathcal{G}$ (řádek 23) [3].

Lemma 12 (Část kontextu odpovídající intervalu $\mathcal{I}_{C,D}$, [3])
Nechť $C\times D\subseteq I$ a $J = I\cap(D^{\downarrow_I}\times C^{\uparrow_I})$, pak $\mathcal{I}_{C,D} = \mathcal{B}(D^{\downarrow_I}, C^{\uparrow_I}, J)$.

Ačkoliv je algoritmus GreEss navržen pro řešení AFP varianty problému, lze jej opět jednoduchou modifikací upravit pro řešení DBP problému (přidání podmínky na velikost množiny $\mathcal{F}$) nebo pro hledání přesné dekompozice (nastavením $\epsilon = 0$) [3].

Algoritmus 6: GreEss

Input: Boolean matrix $I \in \{0,1\}^{n \times m}$, $\epsilon \geq 0$
Output: Set $\mathcal{F}$ of factors

$\mathcal{G} \leftarrow$ ComputeIntervals(I)
$\mathcal{U} \leftarrow \{\langle i, j \rangle \mid I_{ij} = 1\}$
$\mathcal{F} \leftarrow \emptyset$
while $|\mathcal{U}| > \epsilon$ **do**
  $s \leftarrow 0$
  foreach $\langle C, D \rangle \in \mathcal{G}$ **do**
    $J \leftarrow I \cap (D^{\downarrow_I} \times C^{\uparrow_I})$
    $F \leftarrow \emptyset$
    $s_{\langle C,D \rangle} \leftarrow 0$
    while $\exists j \in C^{\uparrow_I} - F \; s.t. \; |(F \cup \{j\})^{\downarrow_J} \times (F \cup \{j\})^{\downarrow_J \uparrow_J} \cap \mathcal{U}| > s_{\langle C,D \rangle}$ **do**
      select j maximizing $|(F \cup \{j\})^{\downarrow_J} \times (F \cup \{j\})^{\downarrow_J \uparrow_J} \cap \mathcal{U}|$
      $E \leftarrow (F \cup \{j\})^{\downarrow_J}$
      $F \leftarrow (F \cup \{j\})^{\downarrow_J \uparrow_J}$
      $s_{\langle C,D \rangle} \leftarrow |E \times F \cap \mathcal{U}|$
    end
    if $s_{\langle C,D \rangle} > s$ **then**
      $\langle E', F' \rangle \leftarrow \langle E, F \rangle$
      $\langle C', D' \rangle \leftarrow \langle C, D \rangle$
      $s \leftarrow s_{\langle C,D \rangle}$
    end
  end
  add $\langle E', F' \rangle$ **to** $\mathcal{F}$
  remove $\langle C', D' \rangle$ **from** $\mathcal{G}$
  $\mathcal{U} \leftarrow \mathcal{U} - E' \times F'$
end
return $\mathcal{F}$

2 Metodologie

Algoritmy Asso, GreConD, GreEss a zde představený Esso (část 3.1) a jeho varianty jsou experimentálně porovnávány na reálných datasetech s využitím poznatků v [8]. Hlavním způsobem měření kvality výsledků algoritmu je *kvalita pokrytí* prvních l faktorů:

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

kde $A \in \{0,1\}^{n \times l}$, $B \in \{0,1\}^{l \times m}$ [8]. Pro určení kvality algoritmu se používá *kvalita faktorizace*:

$$q = 1 - \left(\sum_{j=0}^{l} w_j \frac{E(I, A \circ B)}{\|I\|} \right) \Big/ \left(\sum_{j=0}^{l} w_j \right),$$

kde výpočet váhy w_l se liší podle uvažované varianty problému BMF – pro DBP $w_l = l/k$ a pro AFP $w_l = 1 + (E(I, A \circ B) - \epsilon)/(\|I\| - \epsilon)$ [8]. Z reálných datasetů jsou zde použity: Americas small [9], Chess [10], DBLP [11], Emea [9], Firewall1 [9], Mushroom [10], Paleo [12] a Tic-tac-toe [10], jejichž vybrané vlastnosti jsou uvedeny v Tabulce 1.

Dataset	Dimenze	$\|I\|$	dens(I)	$\|\mathcal{E}(I)\|$	dens($\mathcal{E}(I)$)
Americas small	3477×1587	105 205	0.019	51 258	0.009
Chess	3196×76	118 252	0.487	71 296	0.294
DBLP	6980×19	17 173	0.129	1 601	0.012
Emea	35×3046	7 220	0.068	1 985	0.019
Firewall1	365×709	31 951	0.123	2 787	0.011
Mushroom	8124×119	186 852	0.193	82 965	0.086
Paleo	501×139	3 537	0.051	1 906	0.027
Tic-tac-toe	958×30	9 580	0.333	9 580	0.333

dens$(\cdot)$ vyjadřuje hustotu matice.

Tabulka 1: Reálné datasety

Při testování algoritmu Asso, který je parametrizován vahami w^+ a w^-, je zde zavedeno omezení na $w^+ = w^- = 1$. Navíc platí, že je uváděn ten výsledek pro τ, se kterým bylo dosaženo nejvyšší hodnoty kvality faktorizace q. Testovanými hodnotami parametru τ jsou: .50, .55, .60, .65, .70, .75, .80, .85, .90, .95.

3 Využití $\mathcal{E}(I)$ v algoritmu Asso

Essential část $\mathcal{E}(I)$ matice I lze využít jako kandidátní matici algoritmu Asso, je tomu však nutné přízpůsobit i výpočet ohodnocení (*cover*) jednotlivých kandidátů a konstrukci výsledného faktoru, během něhož bude docházet k aplikaci operátorů $\langle^{\uparrow},{}^{\downarrow}\rangle$. Okamžitě viditelným problémem tohoto přístupu je, že oproti asociační matici, která je čtvercovou maticí a její dimenze jsou rovny počtu sloupců v I, má *essential* část typicky více řádků a tím kandidátů, které je v každé iteraci nutné uvažovat. Bez řádkové klarifikace, tj. odstranění duplicitních řádků, a bez odstranění nulových řádků má $\mathcal{E}(I)$ stejné rozměry jako matice I. Vzhledem k výpočetní náročnosti BMF problému je tento fakt kritický a bude později diskutován a ve výsledném algoritmu patřične adresován.

Následující příklad, převzatý z [4], nastiňuje základní myšlenku a možný postup, který je však v detailech vágní:

Příklad 13

Podtržení v I značí *essential element*.

$X = \{1,2,3,4,5\}$

$Y = \{a,b,c,d,e,f\}$

$$I = \begin{pmatrix} \underline{1} & 0 & 0 & \underline{1} & 0 & 0 \\ 1 & 1 & 1 & 0 & \underline{1} & 1 \\ 0 & \underline{1} & 0 & \underline{1} & 0 & \underline{1} \\ 1 & \underline{1} & \underline{1} & 1 & 0 & 0 \\ 1 & 0 & \underline{1} & 1 & 0 & \underline{1} \end{pmatrix} \qquad C = \mathcal{E}(I) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$k = 1$

$\mathcal{F} = \emptyset$

1 0 0 1 0 0			0 0 0 0 1 0		0 1 0 1 0 1				0 1 1 0 0 0		0 0 1 0 0 1		
1	1	2	0	−1	0	1	0	−1	0 0	−2	0	0	−2
1	0	0	1	1	1	0	1	1	1 1	2	1	1	2
0	1	0	0	−1	1	1	1	3	1 0	0	0	1	0
1	1	2	0	−1	1	1	0	1	1 1	2	1	0	0
1	1	2	0	−1	0	1	1	1	0 1	0	1	1	2
		6		1				**6**		4			4

Nejvyšší skóre je rovno 6, přičemž této hodnoty bylo dosaženo dvěma kandidáty – $c_{0\_}$ a $c_{2\_}$, z nichž bude vybrán $c_{0\_}$, kvůli nižšímu overcover.

1 0 0 1 0 0		
1	1	2
1	0	0
0	1	0
1	1	2
1	1	2

Zatímco Asso by se zajímalo o všechny řádky s ohodnocením větším jak 0, zde budou důležité řádky s maximálním ohodnocením, které se budou dále případně rozšiřovat o další. Zvolené omezení v tomto případě nemá vliv na výslednou množinu řádků/objektů $o = \{1,4,5\}$, z níž aplikací operátorů dojde k sestavení prvního faktoru, tj. $f_1 = \langle o^{\uparrow\downarrow}, o^{\uparrow}\rangle = \langle\{1,4,5\},\{a,d\}\rangle$.

$\mathcal{F} = \mathcal{F} \cup \{\langle\{1,4,5\},\{a,d\}\rangle\}$

Za zmínku stojí, že ačkoliv kandidát $c_{2\_}$ má v tomto případě stejné skóre jako $c_{0\_}$, počet pokrytých pozic odpovídajícím konceptem $\langle\{3\}, \{b, d, f\}\rangle$ by byl nižší než u vybraného f_1. Naopak pro kandidáty $c_{3\_}$ a $c_{4\_}$, které mají nižší skóre, by bylo pokrytí odpovídajícími koncepty, tj. $\langle\{2, 4\}, \{a, b, c\}\rangle$ a $\langle\{2, 5\}, \{a, c, f\}\rangle$, stejné jako u f_1.

$k = 2$

$\mathcal{F} = \{\langle\{1, 4, 5\}, \{a, d\}\rangle\}$

1 0 0 1 0 0		
–	–	0
1	0	0
0	1	0
–	–	0
–	–	0
		0

0 0 0 0 1 0	
0	−1
1	1
0	−1
0	−1
0	−1
	1

0 1 0 1 0 1			
0	–	0	−2
1	0	1	1
1	1	1	3
1	–	0	0
0	–	1	0
			4

0 1 1 0 0 0	
0 0	−2
1 1	2
1 0	0
1 1	2
0 1	0
	4

0 0 1 0 0 1		
0	0	−2
1	1	2
0	1	0
1	0	0
1	1	2
		4

0 1 1 0 0 0
0 0 −2
1 1 2
1 0 0
1 1 2
0 1 0

$o = \{2, 4\}$
$f_2 = \langle\{2, 4\}, \{a, b, c\}\rangle$
Koncept f_2 pokrývá 5 nových pozic ($I_{4,a}$ už je pokryta), alternativní volby ($c_{3\_}$ a $c_{4\_}$) by vedly k pokrytí 3 a 5.

$k = 3$

$\mathcal{F} = \{\langle\{1, 4, 5\}, \{a, d\}\rangle, \langle\{2, 4\}, \{a, b, c\}\rangle\}$

1 0 0 1 0 0		
–	–	0
–	0	−1
0	1	0
–	–	0
–	–	0
		0

0 0 0 0 1 0	
0	−1
1	1
0	−1
0	−1
0	−1
	1

0 1 0 1 0 1			
0	–	0	−2
–	0	1	0
1	1	1	3
–	–	0	−1
0	–	1	0
			3

0 1 1 0 0 0	
0 0	−2
– –	0
1 0	0
– –	0
0 1	0
	0

0 0 1 0 0 1		
0	0	−2
–	1	1
0	1	0
–	0	−1
1	1	2
		3

0 0 1 0 0 1
0 0 −2
– 1 1
0 1 0
– 0 −1
1 1 2

$o = \{5\}$
Množinu objektů lze rozšířit na $o = \{2, 5\}$, případně by bylo možné pracovat s $\{2, 5\}$ okamžitě, jelikož se ve sloupcích vymezených kandidátem jedná o ekvivalentní řádek, který má jednu pozici už překrytou, to znamená, že k hodnotě „–“ by se v tomto ohledu přistupovalo jako k 1.
$f_3 = \langle\{2, 5\}, \{a, c, f\}\rangle$

$k = 4$

$\mathcal{F} = \{\langle\{1,4,5\},\{a,d\}\rangle, \langle\{2,4\},\{a,b,c\}\rangle, \langle\{2,5\},\{a,c,f\}\rangle\}$

1 0 0 1 0 0			0 0 0 0 1 0		0 1 0 1 0 1			0 1 1 0 0 0		0 0 1 0 0 1		
–	–	0	0	−1	0	–	0 −2	0 0	−2	0	0	−2
–	0	−1	1	1	–	0	– −1	– –	0	–	–	0
0	1	0	0	−1	1	1	1 3	1 0	0	0	1	0
–	–	0	0	−1	–	–	0 −1	– –	0	–	0	−1
–	–	0	0	−1	0	–	– −1	0 –	−1	–	–	0
		0		1			**3**		0			0

0 1 0 1 0 1			
0	–	0	−2
–	0	–	−1
1	1	1	3
–	–	0	−1
0	–	–	−1

$o = \{3\}$

$f_4 = \langle\{3\},\{b,d,f\}\rangle.$

$k = 5$

$\mathcal{F} = \{\langle\{1,4,5\},\{a,d\}\rangle, \langle\{2,4\},\{a,b,c\}\rangle, \langle\{2,5\},\{a,c,f\}\rangle, \langle\{3\},\{b,d,f\}\rangle\}$

1 0 0 1 0 0			0 0 0 0 1 0		0 1 0 1 0 1			0 1 1 0 0 0		0 0 1 0 0 1		
–	–	0	0	−1	0	–	0 −2	0 0	−2	0	0	−2
–	0	−1	1	1	–	0	– −1	– –	0	–	–	0
0	–	−1	0	−1	–	–	– 0	– 0	−1	0	–	−1
–	–	0	0	−1	–	–	0 −1	– –	0	–	0	−1
–	–	0	0	−1	0	–	– −1	0 –	−1	–	–	0
		0		**1**			0		0			0

0 0 0 0 1 0		
	0	−1
	1	1
	0	−1
	0	−1
	0	−1

$o = \{2\}$

$f_5 = \langle\{2\},\{a,b,c,e,f\}\rangle$

Výsledkem tedy je:

$$\begin{aligned}\mathcal{F} &= \{\langle\{1,4,5\},\{a,d\}\rangle, \langle\{2,4\},\{a,b,c\}\rangle, \langle\{2,5\},\{a,c,f\}\rangle, \langle\{3\},\{b,d,f\}\rangle,\\ &\quad\ \langle\{2\},\{a,b,c,e,f\}\rangle\}\\ |\mathcal{F}| &= 5\end{aligned}$$

[4] uvádí následující výsledky:

GreConD
$\mathcal{F} = \{\langle\{2,4,5\},\{a,c\}\rangle, \langle\{1,3,4,5\},\{d\}\rangle, \langle\{2,3\},\{b,f\}\rangle, \langle\{1,2,4,5\},\{a\}\rangle,$
$\langle\{2,3,4\},\{b\}\rangle, \langle\{2\},\{a,b,c,e,f\}\rangle, \langle\{2,3,5\},\{f\}\rangle\}$
$|\mathcal{F}| = 7$

GreEss
$\mathcal{F} = \{\langle\{1,4,5\},\{a,d\}\rangle, \langle\{2\},\{a,b,c,e,f\}\rangle, \langle\{3\},\{b,d,f\}\rangle, \langle\{2,4,5\},\{a,c\}\rangle,$
$\langle\{2,3,4\},\{b\}\rangle, \langle\{3,5\},\{d,f\}\rangle\}$
$|\mathcal{F}| = 6$

IterEss
$\mathcal{F} = \{\langle\{2,4\},\{a,b,c\}\rangle, \langle\{1,4,5\},\{a,d\}\rangle, \langle\{3\},\{b,d,f\}\rangle, \langle\{2,5\},\{a,c,f\}\rangle,$
$\langle\{2\},\{a,b,c,e,f\}\rangle\}$
$|\mathcal{F}| = 5$

Tento příklad je však jednoduchý a neprojevují se na něm dostatečně některé problémy, kterými jsou:

1. Již zmíněný problém s ohodnocovací funkcí, tj. skóre neodpovídá vhodnosti použití daného kandidáta. Navíc funkce použitá v příkladu je uměle přejata z algoritmu Asso, např. složka „overcover“ v tomto ohledu nedává smysl, neboť se jedná o aproximaci zdola, a k překrytí tak nemůže dojít.

2. Nelze obecně vzít řádky/objekty s maximální hodnotou pokrytí, protože aplikací operátoru $\cdot^{\uparrow}$ by bylo možné získat prázdnou množinu.

3. Algoritmus pracuje na ose s objekty, která, jak již bylo zmíněno, bývá výrazně vyšší (viz Tabulka 1).

4. Dochází ke zbytečným výpočtům, např. kandidát $c_{1\_}$ není v Příkladu 13 již po svém vybrání nikdy potřeba, přesto se v každé další iteraci stále uvažuje.

Z postupu nastíněném v Příkladu 13 vyšlo několik neuspokojivých variant algoritmu, z nichž některé jsou uvedeny v Příloze A, nicméně vedly k algoritmu popsaném v následující části.

3.1 Esso

Jelikož postupy uvedené v předchozích částech a i Asso samotné hledají, jak co nejlépe pokrýt submatici kontextu I nebo jak na jejím základě pokrýt I, lze pro tyto účely využít postupu použitého v algoritmu GreConD, tj. hledání maximálního obdélníku *greedy* přístupem, a to buď pro konstrukci faktoru nebo zároveň i jako ohodnocovací funkci.

Použití pouze jako způsobu konstrukce faktoru pro kandidáta c, který byl vybrán jako nejlepší, na základě nějaké ohodnocovací funkce, je jednoduché, neboť stačí provést GreConD s $k = 1$ (DBP modifikace) na submatici $\mathcal{U}_c$ a poté na výslednou dvojici $\langle E, F\rangle$ aplikovat operátory indukované kontextem I, to znamená $\langle E^{\uparrow_I\downarrow_I}, E^{\uparrow_I}\rangle$ (nebo jiné ekvivalentní aplikace operátorů na E nebo F, např. $\langle F^{\downarrow_I}, F^{\downarrow_I\uparrow_I}\rangle$). Nevýhodou použití oddělené ohodnocovací funkce je, že neodpovídá výslednému pokrytí nalezeným faktorem – alespoň u zde testovaných funkcí (Příloha A) tomu tak je – a výhoda rychlého výpočtu skóre je tak za cenu horších výsledků. Detaily tohoto postupu jsou uvedeny v Příloze A.

Algoritmus Esso[4] využívá GreConD – konkrétně jeho DBP modifikaci – zároveň jako ohodnocovací funkci a způsob konstrukce výsledného faktoru. V každé iteraci se tak pro každého kandidáta c, tj. řádek matice C, aplikuje GreConD s parametrem k nastaveným na 1, to znamená, že se hledá první maximální obdélník, na matici jím indukovanou, tj. $\mathcal{U}_c$, což také znamená, že operátory používané v GreConD jsou indukované touto submaticí $\mathcal{U}_c$. Pokud je výsledným obdélníkem dvojice $\langle E, F\rangle$, pak ohodnocení příslušného kandidáta lze vypočítat jako $|E^{\uparrow_I\downarrow_I} \times E^{\uparrow_I} \cap \mathcal{U}|$. Je-li tato hodnota maximální (a první nalezená), pak je koncept $\langle E^{\uparrow_I\downarrow_I}, E^{\uparrow_I}\rangle$ přidán jako faktor do $\mathcal{F}$. Tento proces je opakován, dokud není dosaženo limitní podmínky, kterou může být buď:

1. dosažení přesné dekompozice, tj. $|\mathcal{U}| = 0$,
2. nalezení předepsaného počtu k faktorů (DBP varianta), tj. $|\mathcal{F}| = k$,
3. popsání předepsané části matice, vyjádřené přijatelnou chybou ϵ (AFP varianta), tj. $|\mathcal{U}| \leq \epsilon$.

3.1.1 Problém s počtem kandidátů a redukce kandidátní matice C

Představený algoritmus Esso (Algoritmus 7) již zmiňovaný problém s počtem kandidátů ještě umocňuje, neboť použití GreConD s $k = 1$ je náročnější než uvedené ohodnocovací funkce v Příloze A, nicméně způsob, jakým je tento kritický výpočet prováděn a jak jsou řádky kandidátní matice používány, umožňuje následující (potenciální) řešení.

(1) Výpočet pro kandidáta c není nutné v další iteraci provádět, pokud zvolený faktor nezasáhl do $\mathcal{U}_c$. Správnost tohoto tvrzení je zřejmá, neboť výpočet GreConD je deterministický a bez změny $\mathcal{U}_c$ bude výsledek totožný, to

[4]Tak jako je název Asso odvozen od pojmu "association" [1], je Esso jeho modifikací od použité "essential part".

znamená, že místo opakovaného výpočtu by stačilo kontrolovat, zda naposledy přidaný faktor, tj. obdélník $\langle E, F\rangle$, nezasahuje do $\mathcal{U}_c$. Není-li pro kandidáta c potřeba výpočet provádět, je stále nutné ověřit, zda se nezměnilo skóre, neboť to je počítáno z průniku nepokryté části $\mathcal{U}$ a výsledku aplikace operátorů indukovaných kontextem I na obdélník $\langle E, F\rangle$.

(2) Výpočet GreConD pro každého kandidáta je nezávislý a pouze čte sdílená data, kterými je nepokrytá část $\mathcal{U}$ v dané iteraci, což dává prostor pro paralelizaci.

(3) Některé kandidáty c lze z matice C během výpočtu po jejich „vypotřebování“ (viz Příklad 13) vyřadit, podobně jako GreEss vyřazuje již použité intervaly. Co však je dostatečnou podmínkou zde není obsaženo.

Optimalizace (1), (2) a (3) jsou tedy postupy umožněné tím, jak je výpočet prvního maximálního obdélníka v submaticích prováděn, nejsou zde však otestovány a jejich dopady na algoritmus tak nejsou známy.

Optimalizace algoritmu Esso vycházející ze způsobu použití řádků kandidátní matice spočívá v její redukci. Jelikož řádky matice C jsou používány pro výběr submatice, konkrétně tedy výběr těch sloupců, ve kterých má kandidát c hodnotu 1, lze provést jednoduchou redukci, při které dojde jen k malým odchylkám v celkových výsledcích algoritmu a k výraznému urychlení celého procesu.

Algoritmus 7: Esso

Input: Boolean matrix I, number $k \in \mathbb{N}$
Output: set $\mathcal{F}$ of factors

$\mathcal{F} \leftarrow \emptyset$
$C \leftarrow reduce(\mathcal{E}(I))$
$\mathcal{U} \leftarrow \{\langle i, j\rangle \mid I_{ij} = 1\}$
while $|\mathcal{F}| < k$ *and* $|\mathcal{U}| \neq 0$ **do**
  $s \leftarrow 0$
  foreach *row* c *of* C **do**
    $\langle E, F\rangle \leftarrow \text{GreConD}(\mathcal{U}_c, k = 1)$
    $\langle E, F\rangle \leftarrow \langle E^{\uparrow_I\downarrow_I}, E^{\uparrow_I}\rangle$
    $s_{new} \leftarrow |E \times F \cap \mathcal{U}|$
    if $s_{new} > s$ **then**
      $\langle E', F'\rangle \leftarrow \langle E, F\rangle$
      $s \leftarrow s_{new}$
    end
  end
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{\langle E', F'\rangle\}$
  $\mathcal{U} \leftarrow \mathcal{U} - E' \times F'$
end
return $\mathcal{F}$

Prvním způsobem redukce kandidátní matice C je její průchod (jeden), přičemž C je předem:

1. neuspořádaná, tj. *as is* (značeno as-is),
2. uspořádaná dle počtu *essential elements* v řádku (značeno Num),
3. uspořádaná lexikograficky (značeno Lex),
4. uspořádaná nejprve lexikograficky a pak dle počtu *essential elements* v řádku (značeno Lexnum),
5. uspořádaná dle počtu *essential elements* v odpovídajícím řádku $\mathcal{E}(C)$, tedy $\mathcal{E}(\mathcal{E}(I))$ (značeno $\mathcal{E}\mathcal{E}$),

při kterém se *greedy* přístupem vybírají ty řádky, které obsahují alespoň jeden atribut j, který dosud není v kandidátní matici, to znamená, že sloupec j neobsahuje dosud žádnou 1, tj. pro každý řádek row:

$$\text{pokud } \exists j \text{ t.ž. } row_j = 1 \text{ a zároveň } max\, C\_{}_j = 0 \text{ pak přidej } row \text{ do } C.$$

Průchod maticí může být od začátku (značeno ASC) nebo od konce (DESC), což například znamená, že redukce, ve které je použito uspořádání[5] „Num“, bude mít vždy méně nebo stejně řádků při použití pořadí „DESC“ než při „ASC“.

Druhý způsob redukce používá *greedy set-cover* postup s případnými modifikacemi, tj. v každé iteraci se vybere ten řádek row, který:

1. obsahuje nejvíce atributů j, které dosud nejsou obsaženy v C, tj. $row_j = 1$ and $max\, C\_{}_j = 0$ (značeno SC),
2. maximalizuje $n_u - n_c$, kde n_u je počet atributů, které řádek row pokrývá a nejsou dosud obsaženy v C, a n_c je počet atributů, které řádek row sice pokrývá, ale již jsou v C obsaženy, tj.

$$n_u = |\{row_j = 1 \text{ and } max\, C\_{}_j = 0\}|,$$
$$n_c = |\{row_j = 1 \text{ and } max\, C\_{}_j = 1\}|,$$

   přičemž musí platit $n_u > 0$ (značeno SC minus),
3. maximalizuje $n_u \,/\, \|row\|$ (značeno SC divide).

Jak se změní dimenze kandidátní matice po použití těchto redukcí, uvádí Tabulka 2, v níž jsou vyznačeny minimální hodnoty ve sloupcích. Použití libovolné z těchto redukcí je prováděno v Algoritmu 7 na řádku 2 procedurou *reduce*.

[5] Pro všechna uspořádání řádků je použit stabilní třídící algoritmus – konkrétně *merge-sort*.

	Řádky matice C							
Redukce	Americas small	Chess	DBLP	Emea	Firewall1	Mushroom	Paleo	Tic-tac-toe
Původní $\mathcal{E}(I)$	3477	3196	6980	35	365	8124	501	958
Unz($\mathcal{E}(I)$)	210	3196	*19*	35	67	8124	445	958
as-is ASC	166	41	*19*	*34*	63	57	106	17
as-is DESC	170	32	*19*	*34*	63	56	112	12
Num ASC	181	39	*19*	*34*	65	67	127	17
Num DESC	158	29	*19*	*34*	*61*	63	55	12
Lex ASC	182	17	*19*	*34*	65	41	133	17
Lex DESC	154	32	*19*	*34*	*61*	56	84	12
Lexnum ASC	181	39	*19*	*34*	65	67	125	17
Lexnum DESC	155	29	*19*	*34*	*61*	63	54	12
$\mathcal{E}\mathcal{E}$ ASC	168	21	*19*	*34*	62	56	109	17
$\mathcal{E}\mathcal{E}$ DESC	166	34	*19*	*34*	64	67	55	12
SC	*150*	*11*	*19*	*34*	*61*	*34*	*30*	*4*
SC minus	153	*11*	*19*	*34*	*61*	35	39	*4*
SC divide	158	12	*19*	*34*	*61*	*34*	43	*4*

$Unz(\cdot)$ značí řádkovou klarifikaci s odstraněním nulových řádků.

Tabulka 2: Počty řádků v kandidátní matici C po redukci

4 Experimentální výsledky

Tabulka 3 a grafy tvořící Obrázek 5 ukazují, jak představený algoritmus Esso obstojí na reálných datech ve srovnání s algoritmy Asso, GreConD a GreEss, důkladně popsanými v úvodní části 1.3. Nutno poznamenat, že počty potřebných faktorů k dosažení daného pokrytí algoritmy Asso, GreConD a GreEss se mírně liší u zde použitých implementací od těch uváděných v [3]. Důvodem mohou být drobné rozdíly v implementacích a v případě algoritmu Asso navíc výběr výsledků toho τ, pro které bylo dosaženo nejvyšší DBP kvality dle předpisu uvedeném v části 2, což v [2] a [3] ještě nebylo zavedeno.

Na hodnotách pokrytí (*Coverage* v Tabulce 3), na které dosáhne algoritmus Asso, poskytuje Esso podobné výsledky, které se s rostoucí hodnotou pokrytí zlepšují, což je zapříčiněno zanášením chyby překrytím algoritmem Asso, to znamená, že Asso zprvu může dosáhnout nižších hodnot k pro požadované pokrytí, ale za cenu nedosažitelnosti (NA v Tabulce 3) vyšších hodnot pokrytí. Výjimkou je zde dataset Tic-tac-toe, na kterém Asso poskytuje přesnou dekompozici, která je pro pokrytí 95 % a 100 % nejlepší z testovaných algoritmů (o 1 vůči Esso a o 3 vůči GreConD a GreEss), naopak je nejhorší při pokrytí 25 % (o 1 vůči všem). Mimo dataset Tic-tac-toe poskytuje Asso přesnou dekompozici na datasetu DBLP, na kterém si však je na kontrolovaných pokrytích rovno s GreEss a Esso – GreConD zde strádá na 75%, 95% a 100% pokrytí. Algoritmus Asso dosahuje dobrých výsledků ještě na datasetu Paleo, kde maximální dosažené pokrytí je rovno 99.97%, dochází zde tedy k zanesení velmi malého překrytí, konkrétně $E_u = 1$. Nejlepších pokrytí dosahovalo Asso pro hodnoty τ: .75 (na datasetu DBLP), .8 (na datasetech Americas small, Chess, Emea a Tic-tac-toe), .85 (Mushroom), .95 (Firewall1 a Paleo).

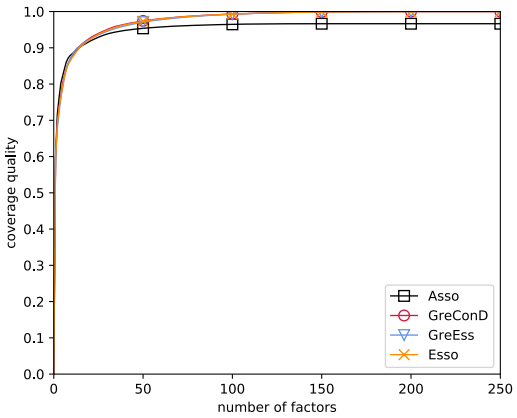
Ve srovnání s testovanými algoritmy vycházejícími z FCA, tj. GreConD a GreEss (IterEss nebylo testováno, počty faktorů potřebných k dosažení předepsané přesnosti jsou pouze převzaty z [4]), jsou výsledky opět velmi těsné a nedochází k výraznějším rozdílům. Zajímavého výsledku bylo dosaženo na datasetu Chess, kde byla přesná dekompozice s nižším počtem faktorů než u novějšího algoritmu IterEss, který používá opakovaný výpočet *essential* části a dosahuje obecně velmi dobrých výsledků [4].

Algoritmus Esso se na testovaných datasetech ukázal být vhodným algoritmem pro oba pohledy BMF problému (DBP i AFP), neboť jím dosahovaná kvalita pokrytí se většinou pohybuje mezi těmi poskytovanými algoritmy GreConD a GreEss. Jsou však pozorovatelné případy, kdy je nejnižší z těchto tří algoritmů, např. pro $k < 10$ na datasetu Firewall1 nebo Mushroom pro $k = 29$ až $k = 78$.

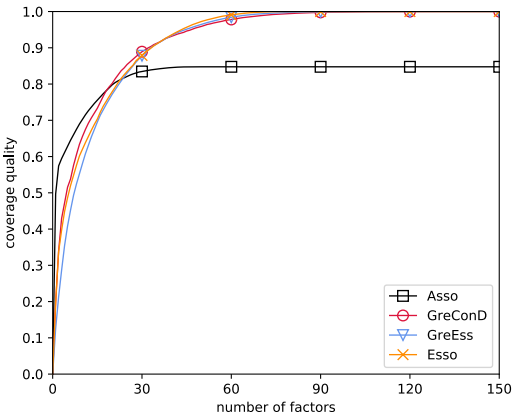
Uváží-li se metody redukce kandidátní matice v algoritmu Esso (Tabulka 4 a grafy na Obrázku 6, popř. 9 v Příloze B), je možné některé výsledky dokonce vylepšit, a smazat tak nebo zmenšit některé z rozdílů vzhledem k algoritmu GreEss, např. u datasetů Mushroom, Paleo a Chess. To znamená, že dochází nejen k výraznému snížení počtu kandidátů (Tabulka 2), které je během výpo-

čtu potřeba testovat, a tím ke zrychlení hledání jednotlivých faktorů, ale může dojít i ke zlepšení samotných faktorizací. Ne všechny představené redukce jsou však dobré, například Num ASC a Lexnum ASC vede k horší faktorizaci datasetu Mushroom, ačkoliv vylepšují Chess, redukce $\mathcal{EE}$ ASC zhoršuje výsledek na Americas small a Mushroom, naopak vylepšuje Paleo, SC má zase negativní efekt na Americas small a Chess, SC minus na Chess. U některých datasetů, zde DBLP, Emea, Firewall1, se zase redukce v rozdílném pokrytí neprojevuje vůbec, což je také příznivé. Na datasetu Americas small dochází u všech redukcí k mírnému zhoršení pro pokrytí 100% (v rozmezí 1–5 faktorů), v některých případech i na 95% (o 1 faktor) a jedná se tak o jediný z testovaných datasetů, na kterém došlo u všech redukcí pouze ke zhoršení výsledků na kontrolovaných hodnotách pokrytí.

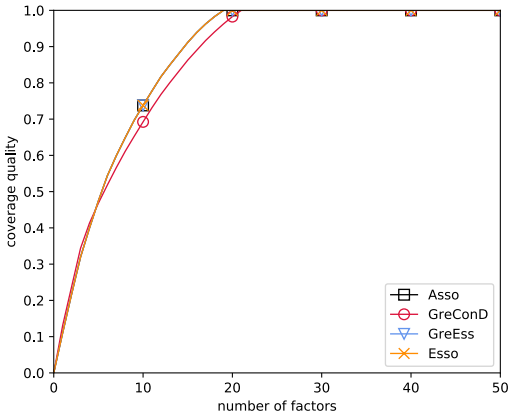
Z důvodu podobnosti kvality pokrytí jednotlivých variant Esso s redukcí kandidátní matice a jejich počtu, jsou zde uváděny výsledky jen vybraných variant a datasetů, konkrétně se jedná o: as-is ASC, Num DESC, Lex ASC, Lexnum DESC, $\mathcal{EE}$ DESC a SC divide na datasetech Chess a Mushroom. Pro úplnost jsou výsledky všech variant redukcí a datasetů uvedeny v Příloze B.



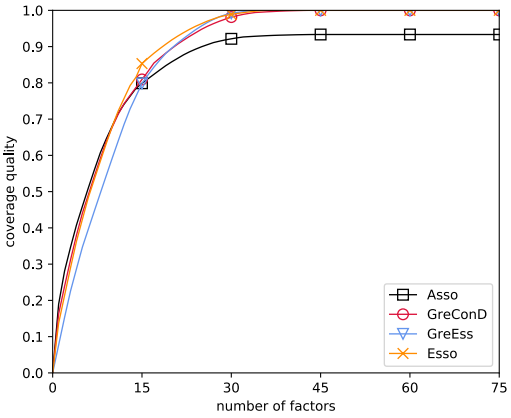
(a) Americas small



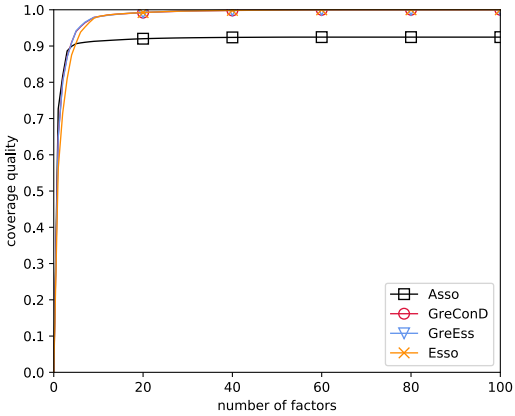
(b) Chess



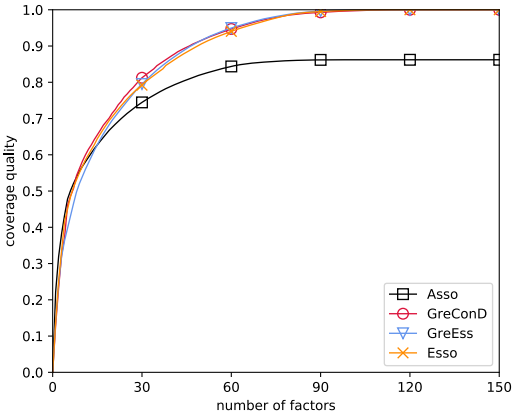
(c) DBLP



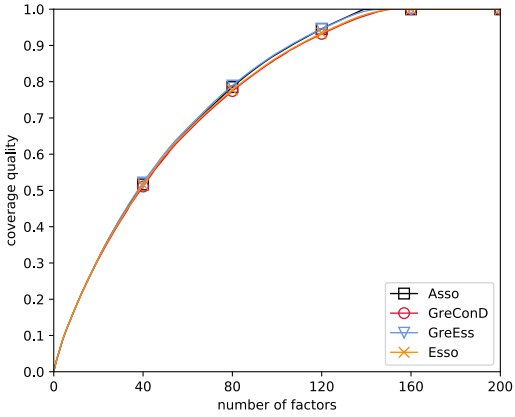
(d) Emea



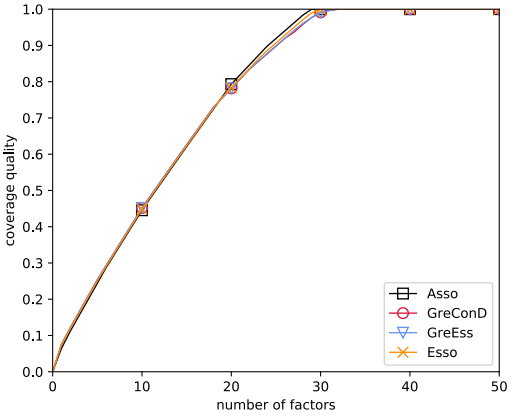
(e) Firewall1



(f) Mushroom



(g) Paleo



(h) Tic-tac-toe

Obrázek 5: Kvalita pokrytí Esso na reálných datech

Dataset	Coverage	Počet faktorů potřebných k předepsané přesnosti				
		GreConD	GreEss	IterEss[a]	Asso	Esso
Americas small	25%	1	1	—	1	1
	50%	1	1	—	1	1
	75%	4	4	—	3	4
	95%	30	33	32	43	31
	100%	197	191	184	NA	194
Chess	25%	2	3	—	1	2
	50%	5	8	—	2	6
	75%	16	19	—	15	18
	95%	47	45	43	NA	44
	100%	124	110	98	NA	81
DBLP	25%	3	3	—	3	3
	50%	6	6	—	6	6
	75%	12	11	—	11	11
	95%	19	17	—	17	17
	100%	21	19	—	19	19
Emea	25%	3	4	—	2	3
	50%	7	9	—	6	7
	75%	13	14	—	13	12
	95%	26	25	—	NA	24
	100%	43	32	—	NA	36
Firewall1	25%	1	1	—	1	1
	50%	1	1	—	1	1
	75%	2	2	—	2	3
	95%	6	6	—	NA	7
	100%	66	64	—	NA	65
Mushroom	25%	3	3	—	2	2
	50%	7	9	—	6	7
	75%	24	26	—	32	25
	95%	62	61	57	NA	64
	100%	120	105	99	NA	113
Paleo	25%	16	15	—	15	16
	50%	39	38	—	39	39
	75%	76	73	—	74	75
	95%	127	122	122	122	126
	100%	151	145	139	NA	151
Tic-tac-toe	25%	5	5	—	6	5
	50%	12	12	—	12	12
	75%	19	19	—	19	19
	95%	28	28	—	27	28
	100%	32	32	—	29	30

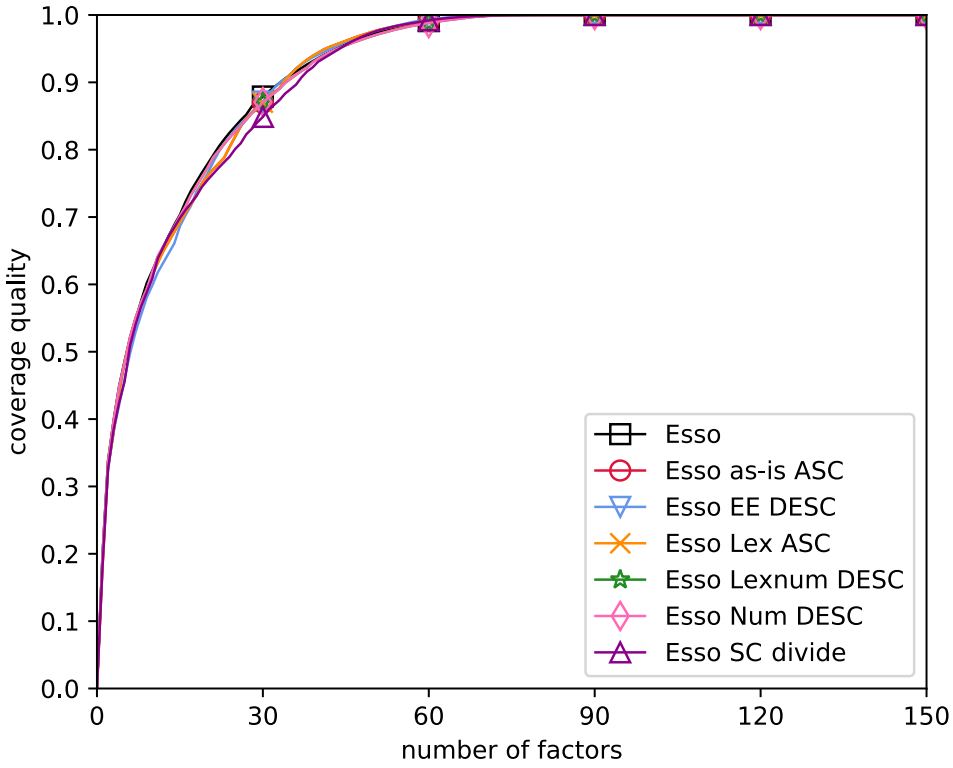
[a] výsledky převzaté z [4].
— nedostupné z důvodu nepořízených výsledků.

Tabulka 3: Potřebný počet faktorů pro požadované pokrytí pro Esso na reálných datech

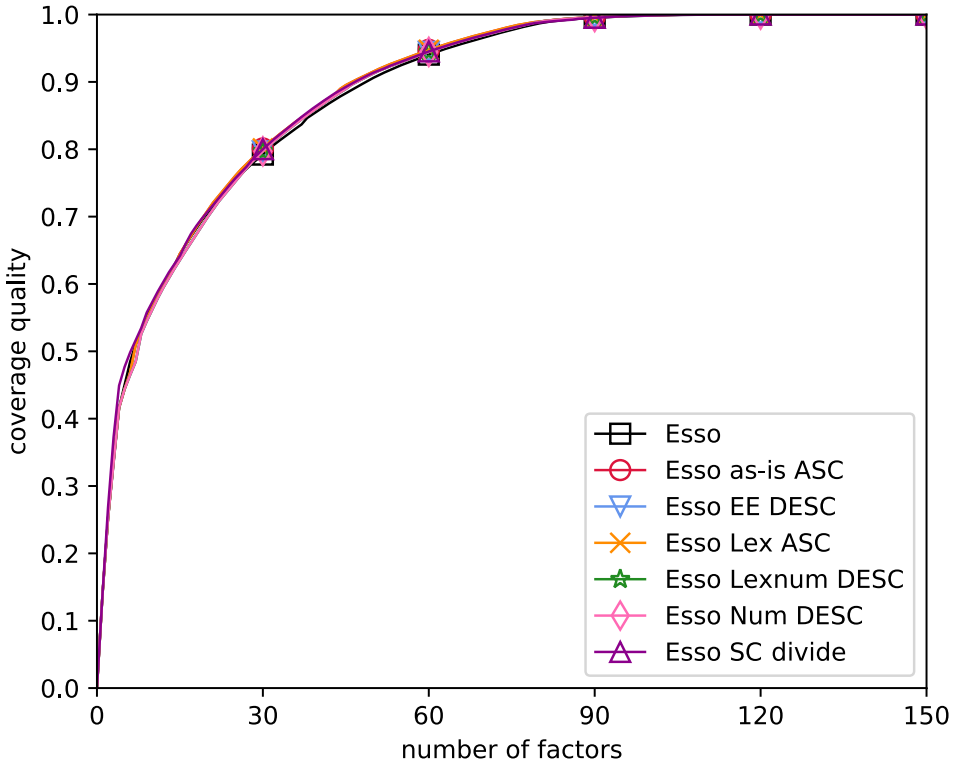
		Počet faktorů potřebných k dosažení předepsané přesnosti																
Dataset	Coverage	GreCond	GreEss	Asso	Esso	as-is ASC	as-is DESC	Num ASC	Num DESC	Lex ASC	Lex DESC	Lexnum ASC	Lexnum DESC	$\mathcal{EE}$ ASC	$\mathcal{EE}$ DESC	SC	SC minus	SC divide
Americas small	25%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	50%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	75%	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	95%	30	33	63	31	31	32	31	32	31	32	31	32	31	31	32	32	32
	100%	197	191	NA	194	195	195	195	196	197	196	196	196	199	196	198	194	195
Chess	25%	2	3	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	50%	5	8	2	6	6	7	7	6	6	7	7	6	6	7	7	7	6
	75%	16	19	15	18	20	19	19	19	20	19	19	19	19	19	21	23	20
	95%	47	45	NA	44	42	48	42	44	42	48	42	44	44	43	49	50	44
	100%	124	110	NA	81	78	82	76	77	78	82	76	77	80	76	82	79	79
DBLP	25%	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	50%	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	75%	12	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
	95%	19	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17
	100%	21	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
Emea	25%	3	4	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	50%	7	9	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7
	75%	13	14	13	12	12	12	12	12	12	12	12	12	12	12	12	12	12
	95%	26	25	NA	24	24	24	24	24	24	24	24	24	24	24	24	24	24
	100%	43	32	NA	36	36	36	36	36	36	36	36	36	36	36	36	36	36

Dataset	Coverage	GreConD	GreEss	Asso	Esso	as-is ASC	as-is DESC	Num ASC	Num DESC	Lex ASC	Lex DESC	Lexnum ASC	Lexnum DESC	$\mathcal{EE}$ ASC	$\mathcal{EE}$ DESC	SC	SC minus	SC divide
Firewall1	25%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	50%	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	75%	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	95%	6	6	NA	7	7	7	7	7	7	7	7	7	7	7	7	7	7
	100%	66	64	NA	65	65	65	65	65	65	65	65	65	65	65	65	65	65
Mushroom	25%	3	3	2	2	2	3	3	2	2	3	3	2	3	2	2	2	2
	50%	7	9	6	7	7	9	9	8	7	9	9	8	9	8	7	7	7
	75%	24	26	32	25	24	25	27	25	24	25	27	25	27	25	26	25	25
	95%	62	61	NA	64	61	62	66	62	61	62	66	62	66	62	62	63	62
	100%	120	105	NA	113	110	111	115	111	110	111	115	110	115	110	110	111	112
Paleo	25%	16	15	15	16	16	16	15	16	15	16	15	16	16	16	16	16	15
	50%	39	38	39	39	39	39	38	39	38	39	38	39	39	39	39	39	38
	75%	76	73	74	75	74	75	75	74	74	75	74	74	75	75	74	74	73
	95%	127	122	122	126	124	126	124	125	124	124	123	125	123	125	123	123	122
	100%	151	145	NA	151	147	146	145	146	146	148	144	146	144	145	143	144	143
Tic-tac-toe	25%	5	5	6	5	5	6	5	6	5	6	5	6	5	6	6	6	6
	50%	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
	75%	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
	95%	28	28	27	28	27	27	27	27	27	27	27	27	27	27	27	27	27
	100%	32	32	29	30	30	30	30	30	30	30	30	30	30	30	29	29	29

Tabulka 4: Potřebný počet faktorů pro požadované pokrytí pro Esso s redukcí kandidátní matice na reálných datech



(a) Chess



(b) Mushroom

Obrázek 6: Kvalita pokrytí Esso s redukcí kandidátní matice na reálných datech

Závěr

Esso, algoritmus představený v této práci, využívá abstrakci algoritmu Asso, která pro každého kandidáta (řádek) matice C vybere jím indukovanou submatici. Kandidátovi je na základě této submatice přiřazeno ohodnocení a pokud je kandidát vybrán, slouží tato matice i pro konstrukci faktoru. Esso pro obojí využívá greedy přístup, který byl představen v algoritmu GreConD. Jako kandidátní matice je používána essential část matice I, značeno $\mathcal{E}(I)$. Jelikož ale tato matice má stejný počet řádků jako původní I, byly pro účely tohoto algoritmu navrženy jednoduché způsoby její redukce s cílem snížit množství uvažovaných a testovaných kandidátů a tím urychlit faktorizaci. Přestože navržené redukce jsou převážně jednoduché, byly výsledky algoritmu Esso po jejich aplikaci na kandidátní matici podobné jako bez redukce a na testovaných datasetech nedošlo k výraznějším odchylkám v kvalitě pokrytí a v některých případech dokonce došlo k nalezení lepších faktorizací. Mimo redukce kandidátní matice byly navrženy další potenciální optimalizace, které by spolu se sofistikovanějšími způsoby redukce mohly být prostorem pro zlepšení představeného algoritmu.

V experimentu na reálných datech se Esso zdá být obstojným algoritmem pro obě varianty BMF problému – AFP a DBP. Výhodou algoritmu Esso oproti algoritmu Asso je, že se jedná o aproximaci zdola, a je tak schopné dosáhnout přesné dekompozice, což u algoritmu Asso není obecně možné z důvodu zanášení chyby překrytím. Dále také odpadá nutnost specifikovat dodatečné parametry mimo k a ϵ, které jsou dány variantou problému (DBP a AFP resp.). Oproti tomu, Asso je parametrizováno vahami pro oba typy chyby, tj. w^- a w^+, a τ, které je používáno pro určení potřebné síly (hodnoty confidence) asociačních pravidel mezi sloupci. Esso je tak ve srovnání s algoritmem Asso obecnějším BMF algoritmem, neboť může být použito jak pro DBP tak AFP, a lze jej také snáze použít z důvodu absence dodatečných parametrů. Esso lze také lehce implementovat, protože využívá jednoduchou strukturu algoritmu Asso.

Conclusions

Esso, the algorithm introduced in this work, utilizes an abstraction of the Asso algorithm, which assigns a score to the submatrices induced by individual rows (called candidates) of the candidate matrix C. The submatrix of the best candidate is then used to construct the next factor. However, Esso uses a greedy approach, which was introduced in the GreConD algorithm, for both the scoring and searching. The essential part of matrix I, noted $\mathcal{E}(I)$, is used as the candidate matrix, but this matrix has the same number of rows as the original I, which lead to the construction of simple reduction methods for the purpose of Esso. These methods minimize the number of considered and tested candidates when searching for individual factors and consequently speed up decomposition. Even though the proposed reductions are simple, the results of Esso on the tested datasets after the reduction of the candidate matrix were similar to those without the reduction and deviated only slightly in terms of coverage quality and, in some cases, provided even better coverage. Additionally, other potential optimization methods for Esso, besides the reduction of candidate matrix, were proposed in this work and, together with the more sophisticated methods of reduction, could provide room for improvement.

In an experiment with real-world datasets, Esso seems to be an algorithm dealing efficiently with both variants of the BMF problem – AFP and DBP. In comparison with Asso, Esso has the advantage of being a from-below approximation, thus being able to achieve an exact decomposition, which Asso is generally unable to accomplish due to the overcovering error. In Esso, the necessity to specify additional parameters besides those prescribed by the variant of the BMF problem vanishes as well, as the algorithm is parameterized only by the number of factors k or tolerable error ϵ for DBP and AFP respectively. In contrast, Asso is always parameterized by the weights of both types of error w^- and w^+ and τ used to determine the necessary strength of association rule. To conclude, in comparison with Asso, Esso is a more general-purpose BMF algorithm, as it can be used for both DBP and AFP. It is also easier to use due to the lack of additional parameters while benefitting from the simple structure of Asso, which makes it easy to implement.

A Další testované způsoby ohodnocování kandidátů a hledání faktorů

Mimo představený algoritmus Esso byly vyzkoušeny ještě jiné způsoby ohodnocování submatic $\mathcal{U}_c$ a hledání faktorů. Jelikož však nedosahují dobrých výsledků jsou uváděny až mimo hlavní text práce. O ohodnocovacích funkcích padla zmínka již v části 3.1 v případě použití GreConD pouze jako způsobu hledání faktoru. Jelikož tyto varianty vedly později k formulování algoritmu Esso, jsou označovány jako Pesso (*pre-* nebo *proto*Esso).

Mimo *cover* funkci algoritmu Asso [1], jejíž použití je značeno příponou *ASSO*, byly vyzkoušeny „naivní" funkce, které jsou však velmi chybné ve smyslu, že jimi přiřazovaná hodnota submaticím neodpovídá vhodnosti daného kandidáta (obdobně jako bylo zmíněno v Příkladě 13 pro $k = 1$ při použití Asso *cover*); těmito funkcemi jsou:

1. Suma nově pokrytých pozic, značeno příponou *SUM*.
2. Suma nově pokrytých *essential* pozic, značeno *ESUM*.
3. Hustota submatice $\mathcal{U}_c$, značeno *DENS*.
4. Hustota submatice nepokryté části $\mathcal{E}(I)_c$, značeno *EDENS*.
5. Počet nově pokrytých pozic na *essential element* kandidáta c, tj.:

$$\frac{|\{\langle i,j\rangle : I_{ij}, (A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij} = 0, c_j = 1\}|}{\|c\|},$$

   což však lze ještě zjednodušit na:

$$\frac{|\{\langle i,j\rangle : \mathcal{U}_{ij} = 1, c_j = 1\}|}{\|c\|}.$$

   Tato ohodnocovací funkce je značena *ECOV*.

Pro „nejlepšího" kandidáta vybraného, na základě jedné z těchto funkcí, byly testovány dva způsoby konstrukce faktoru:

1. Aplikace GreConD na submatici $\mathcal{U}_c$ indukovanou daným kandidátem c, to znamená, že šipkové operátory jsou při hledání prvního faktoru indukovány touto submaticí. Na vybraný obdélník jsou před přidáním mezi faktory aplikovány operátory indukované kontextem I. Použití tohoto způsobu je značeno příponou GreConD, tj. Pesso-GreConD.
2. Varianta Pesso-CE se zajímá o řádky v submatici $\mathcal{U}_c$, které mají maximální hodnotu pokrytí, tj. obsahují nejvíce prvků (suma řádku), a jsou tak považovány za potenciální základ nebo jádro faktoru (přípona CE je

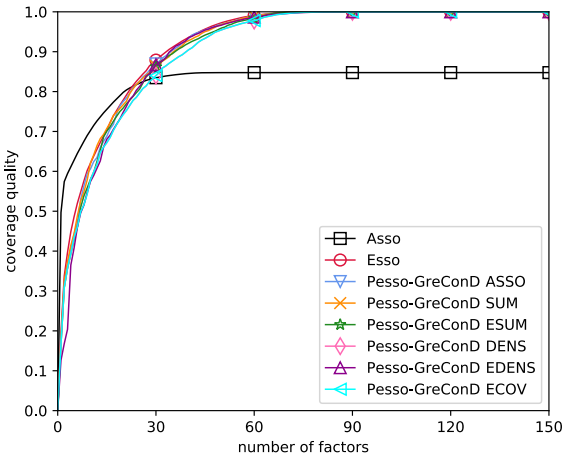
zkratka pro *core extension*). Každé z těchto jader je následně *greedy* způsobem rozšiřováno o další řádky, přičemž jsou vždy přidány jemu ekvivalentní řádky v submatici $\mathcal{U}_c$, to znamená řádky, které mají v submatici 1 na stejných pozicích. Poté jsou aplikovány šipkové operátory indukované nepokrytou částí $\mathcal{U}$ (celou), alternativně by bylo možné použít i operátory indukované I, ale tato varianta ve většině případů poskytovala mírně horší výsledky (v jednotkách faktorů). Na takto nalezený výsledek jsou ještě před přidáním mezi faktory použity šipkové operátory, tentokrát indukované původním kontextem I. Tato varianta je tak nejblíže postupu nastíněném v Příkladu 13.

Srovnání variant Pesso s výsledky algoritmů Asso a Esso (bez redukce kandidátní matice) popisuje Tabulka 5 a grafy na Obrázku 7 a 8. Ani jedna varianta Pesso však není konzistentní, ačkoliv na datasetech Tic-tac-toe, Chess, DBLP (s výjimkou ohodnocovací funkce EDENS a ESUM) nebo i Paleo jsou výsledky blízké těm algoritmu Esso, na datasetech Firewall1 a Emea už jsou vidět špatné volby faktorů související se zmiňovanými problémy ohodnocovacích funkcí. Navíc s výjimkou datasetu Paleo se standardní ohodnocovací funkce algoritmu Asso stále jeví být nejlepší, což není překvapivé vzhledem k jednoduchosti zde testovaných alternativních funkcí, které byly již od začátku označovány jako „naivní“ právě z tohoto důvodu. Hledání lepších ohodnocovacích funkcí nebylo prováděno, neboť varianta použití GreConD postupu jak pro hledání faktoru, tak pro účel ohodnocení kandidátů, tedy algoritmus Esso, měla slibnější výsledky a společně s navrženou redukcí kandidátní matice 3.1.1 jich dosahovala i v příznivém čase.

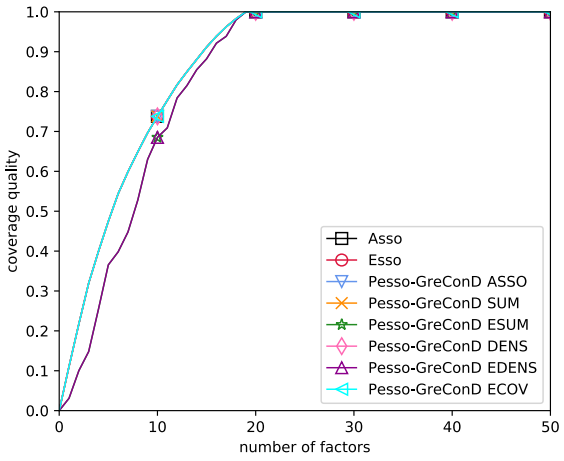
Dataset	Coverage	Počet faktorů potřebných k předepsané přesnosti													
				Pesso-GreConD						Pesso-CE					
		Asso	Esso	ASSO	SUM	ESUM	DENS	EDENS	ECOV	ASSO	SUM	ESUM	DENS	EDENS	ECOV
Chess	25%	1	2	2	2	2	2	4	2	2	2	2	2	2	2
	50%	2	6	7	7	7	8	8	8	6	6	6	6	6	6
	75%	15	18	19	19	19	21	21	21	19	19	18	20	20	20
	95%	NA	44	44	45	48	49	44	49	45	46	46	46	47	46
	100%	NA	81	78	80	79	85	87	85	86	85	84	88	86	88
DBLP	25%	3	3	3	3	4	3	4	3	3	3	4	3	4	3
	50%	6	6	6	6	8	6	8	6	6	6	8	6	8	6
	75%	11	11	11	11	12	11	12	11	11	11	12	11	12	11
	95%	17	17	17	17	18	17	18	17	17	17	18	17	18	17
	100%	19	19	19	19	19	19	19	19	19	19	19	19	19	19
Emea	25%	2	3	5	5	5	9	10	9	5	5	5	5	10	5
	50%	6	7	12	12	12	15	20	15	12	12	12	14	18	14
	75%	13	12	15	15	21	27	27	27	15	15	21	27	28	27
	95%	NA	24	28	28	28	34	33	34	26	26	27	35	34	35
	100%	NA	36	35	35	35	37	35	37	36	36	36	38	37	38
Firewall1	25%	1	1	1	1	2	1	3	1	1	1	2	1	3	1
	50%	1	1	1	1	2	1	3	1	1	1	2	1	3	1
	75%	2	3	3	3	4	6	5	6	3	3	4	6	5	6
	95%	NA	7	7	7	11	26	50	26	7	7	11	25	50	25
	100%	NA	65	65	65	65	67	67	67	65	65	65	66	67	66

Dataset	Coverage	Počet faktorů potřebných k předepsané přesnosti													
				Pesso-GreConD						Pesso-CE					
		Asso	Esso	ASSO	SUM	ESUM	DENS	EDENS	ECOV	ASSO	SUM	ESUM	DENS	EDENS	ECOV
Paleo	25%	15	16	17	17	19	16	17	16	16	17	18	16	18	16
	50%	39	39	40	43	46	39	44	39	40	42	43	39	44	39
	75%	74	75	78	79	88	77	79	77	79	79	83	77	79	77
	95%	122	126	126	127	130	123	125	123	128	127	132	123	125	123
	100%	NA	151	148	147	145	144	144	144	152	147	149	144	144	144
Tic-tac-toe	25%	6	5	5	5	5	5	5	5	5	5	5	5	5	5
	50%	12	12	12	12	12	12	12	12	12	12	12	12	12	12
	75%	19	19	19	19	19	19	19	19	19	19	19	19	19	19
	95%	27	28	28	28	28	28	28	28	28	28	28	28	28	28
	100%	29	30	30	31	31	31	31	31	30	30	30	30	30	30

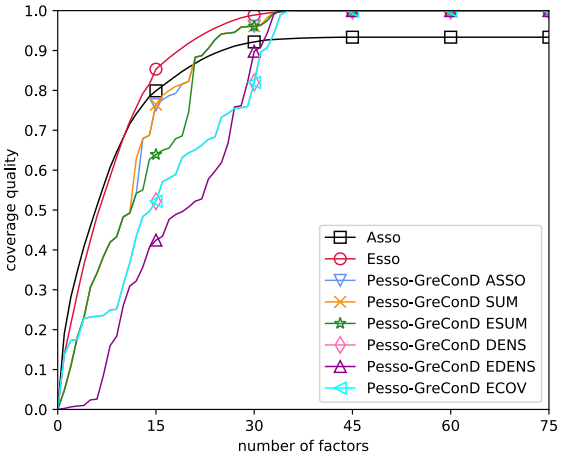
Tabulka 5: Potřebný počet faktorů pro požadované pokrytí pro Pesso varianty na reálných datech



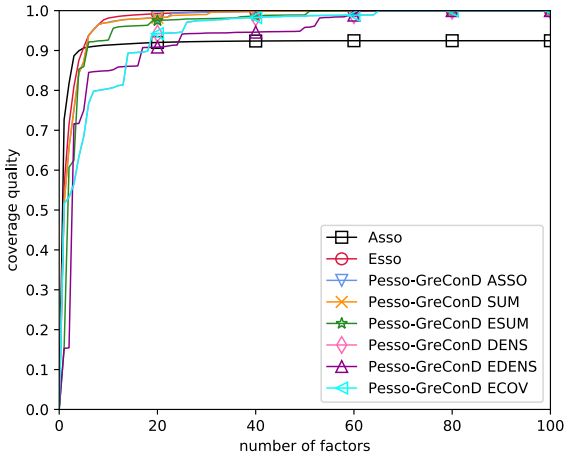
(a) Chess



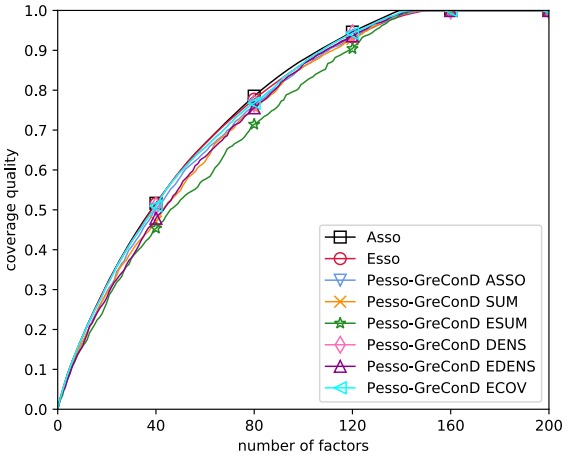
(b) DBLP



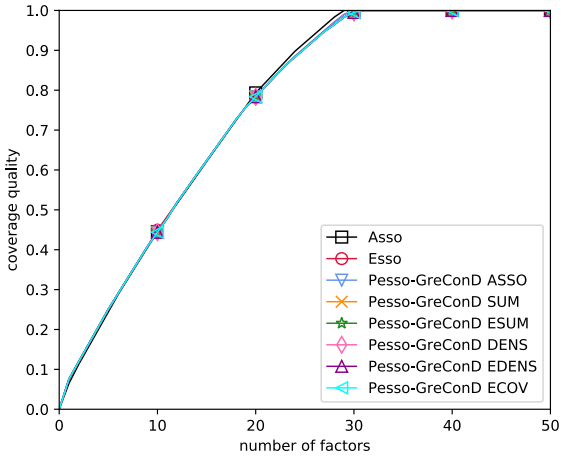
(c) Emea



(d) Firewall1

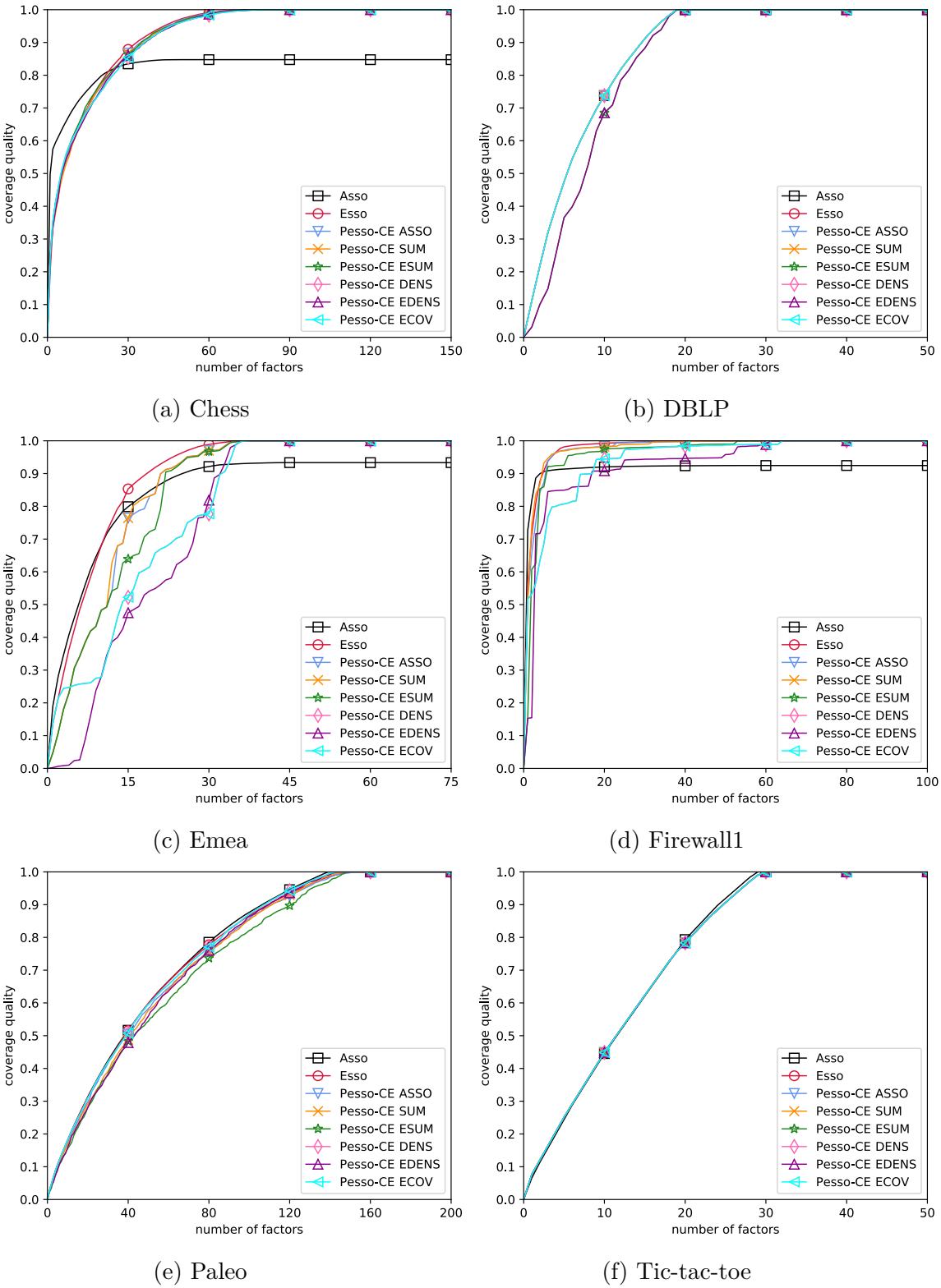


(e) Paleo



(f) Tic-tac-toe

Obrázek 7: Kvalita pokrytí Pesso s využitím GreConD na reálných datech



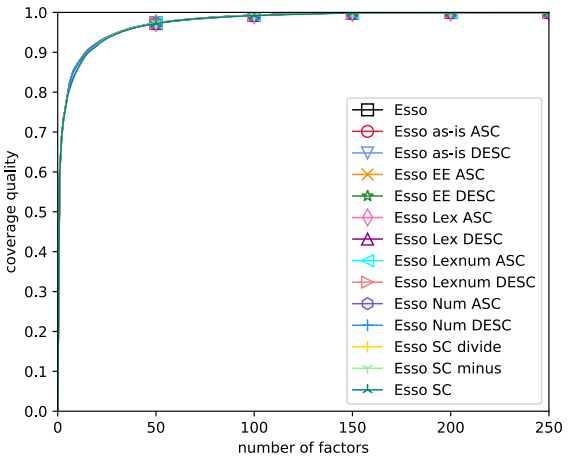
(a) Chess (b) DBLP

(c) Emea (d) Firewall1

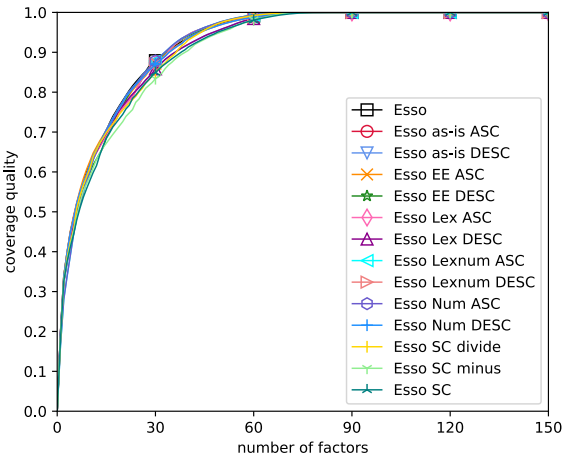
(e) Paleo (f) Tic-tac-toe

Obrázek 8: Kvalita pokrytí Pesso s *core extension* na reálných datech

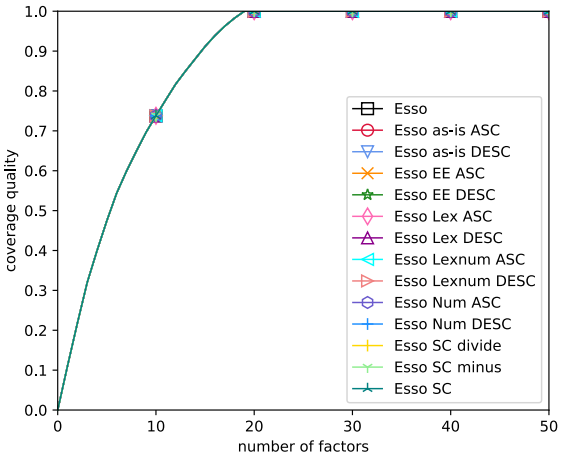
B Grafy kvality pokrytí Esso s redukcí kandidátní matice



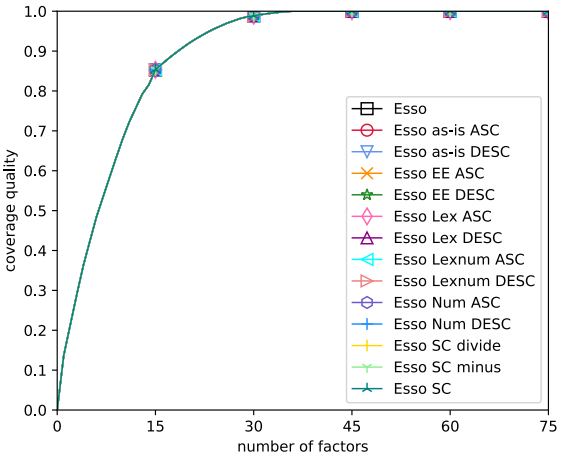
(a) Americas small



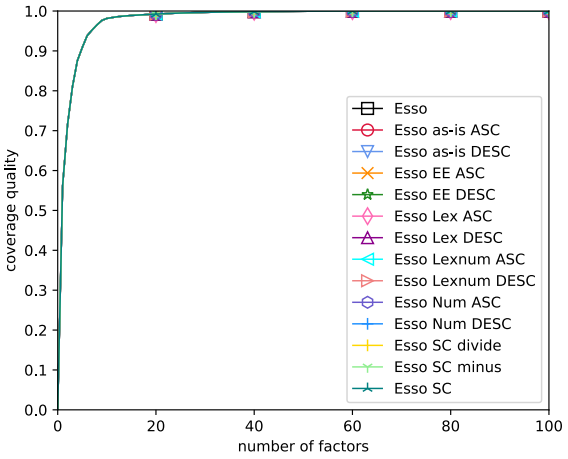
(b) Chess



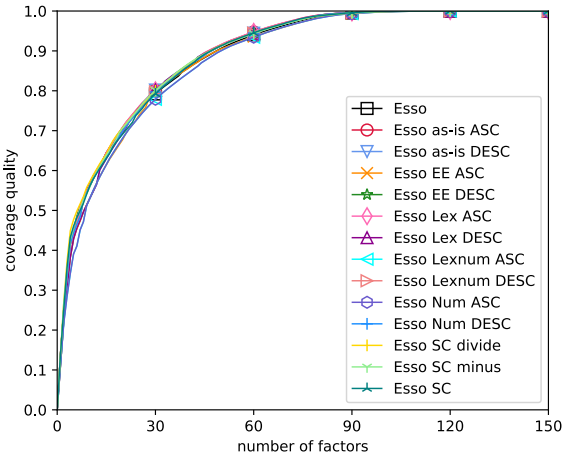
(c) DBLP



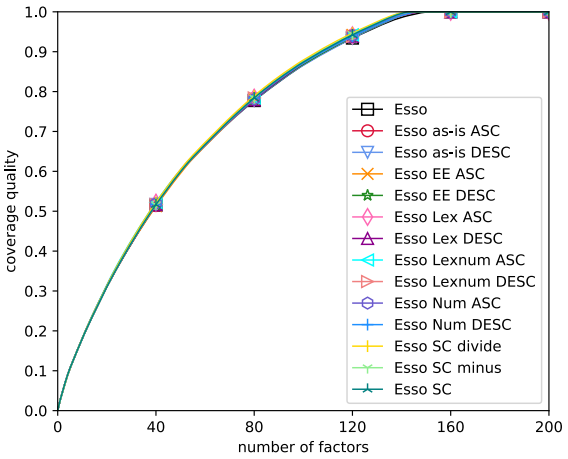
(d) Emea



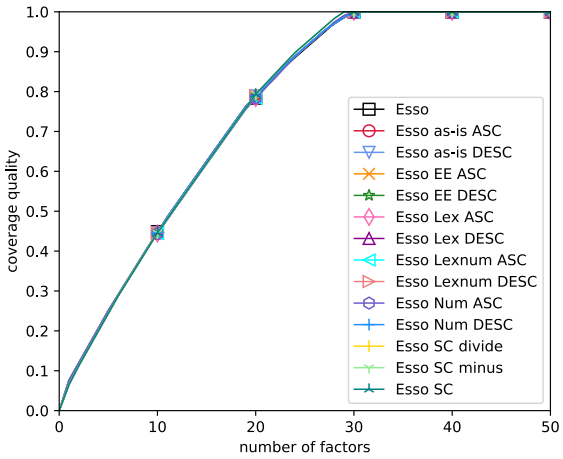
(e) Firewall1



(f) Mushroom



(g) Paleo



(h) Tic-tac-toe

Obrázek 9: Kvalita pokrytí Esso s redukcí kandidátní matice na reálných datech

C Obsah přiloženého CD/DVD

datasets/ Použité datasety.

doc/ Text práce (PDF) a soubory s přílohami potřebnými pro jeho vygenerování.

results/ Faktorizace testovaných datasetů algoritmy Asso, GreConD, GreEss, Esso a Pesso.

src/ Implementace testovaných algoritmů.

Literatura

[1] MIETTINEN, P.; MIELIKÄINEN, T.; GIONIS, A.; DAS, G.; MANNILA, H. The Discrete Basis Problem. *IEEE Transactions on Knowledge and Data Engineering*. 2008, roč. 20, č. 10, s. 1348–1362. Dostupný také z: ⟨http://dx.doi.org/10.1109/TKDE.2008.53⟩.

[2] BELOHLAVEK, Radim; VYCHODIL, Vilem. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *Journal of Computer and System Sciences*. 2010, roč. 76, č. 1, s. 3–20. Special Issue on Intelligent Data Analysis. Dostupný také z: ⟨http://www.sciencedirect.com/science/article/pii/S0022000009000415⟩. ISSN 0022-0000.

[3] BELOHLAVEK, Radim; TRNECKA, Martin. From-Below Approximations in Boolean Matrix Factorization: Geometry and New Algorithm. *Journal of Computer and System Sciences*. 2013, roč. 81. Dostupný také z: ⟨http://dx.doi.org/10.1016/j.jcss.2015.06.002⟩.

[4] BELOHLAVEK, Radim; OUTRATA, Jan; TRNECKA, Martin. Factorizing Boolean matrices using formal concepts and iterative usage of essential entries. *Information Sciences*. 2019, roč. 489, s. 37–49. Dostupný také z: ⟨http://www.sciencedirect.com/science/article/pii/S0020025519301902⟩. ISSN 0020-0255.

[5] BELOHLAVEK, R. *Introduction to formal concept analysis*. Dostupný také z: ⟨http://belohlavek.inf.upol.cz/vyuka/IntroFCA.pdf⟩.

[6] TRNECKA, Martin; TRNECKOVA, Marketa. Data reduction for Boolean matrix factorization algorithms based on formal concept analysis. *Knowledge-Based Systems*. 2018, roč. 158, s. 75–80. Dostupný také z: ⟨http://www.sciencedirect.com/science/article/pii/S0950705118302594⟩. ISSN 0950-7051.

[7] TRNECKA, M. *Decompositions of Matrices with Relational Data: Fundations and Algorithms*. 2016.

[8] BELOHLAVEK, R.; OUTRATA, J.; TRNECKA, M. How to assess quality of BMF algorithms? In. *2016 IEEE 8th International Conference on Intelligent Systems (IS)*. 2016, s. 227–233. Dostupný také z: ⟨http://dx.doi.org/10.1109/IS.2016.7737426⟩.

[9] ENE, Alina; HORNE, William; MILOSAVLJEVIC, Nikola aj. Fast Exact and Heuristic Methods for Role Minimization Problems. In. *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*. Estes Park, CO, USA: ACM, 2008, s. 1–10. SACMAT '08. Dostupný také z: ⟨http://doi.acm.org/10.1145/1377836.1377838⟩. ISBN 978-1-60558-129-3.

[10] LICHMAN, M. *UCI Machine Learning Repository*. 2013. Dostupný z: ⟨http://archive.ics.uci.edu/ml⟩.

[11] MIETTINEN, P. *Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms*. 2009.

[12] *NOW public release 030717*. Dostupný z: ⟨http://www.helsinki.fi/science/now/⟩.