Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

# DIPLOMOVÁ PRÁCE

Vizualizace Booleovské dekompozice matic

2020

Bc. Tomáš Vlk

Vedoucí práce: doc. Mgr. Jan Outrata, Ph.D.

Studijní obor: Informatika, prezenční forma

**Bibliografické údaje**

| | |
|---|---|
| Autor: | Bc. Tomáš Vlk |
| Název práce: | Vizualizace Booleovské dekompozice matic |
| Typ práce: | diplomová práce |
| Pracoviště: | Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci |
| Rok obhajoby: | 2020 |
| Studijní obor: | Informatika, prezenční forma |
| Vedoucí práce: | doc. Mgr. Jan Outrata, Ph.D. |
| Počet stran: | 52 |
| Přílohy: | 1 CD/DVD |
| Jazyk práce: | český |

**Bibliograhic info**

| | |
|---|---|
| Author: | Bc. Tomáš Vlk |
| Title: | Visualisation of Boolean matrix decomposition |
| Thesis type: | master thesis |
| Department: | Department of Computer Science, Faculty of Science, Palacký University Olomouc |
| Year of defense: | 2020 |
| Study field: | Computer Science, full-time form |
| Supervisor: | doc. Mgr. Jan Outrata, Ph.D. |
| Page count: | 52 |
| Supplements: | 1 CD/DVD |
| Thesis language: | Czech |

## Anotace

*Práce se zabývá vizualizací Booleovské dekompozice matic na vstupních datech. Popisuje přesnou podobu vizualizace a implementuje ji do multiplatformní aplikace, umožňující vhled do jednotlivých kroků dekompozice. Uživateli jsou dány možnosti, jak interaktivně měnit pořadí kroků, sledovat efektivitu, nebo hledat zajímávé informace ve vstupních datech.*

## Synopsis

*The thesis deals with visualisation of Boolean matrix decomposition directly on input data. It describes exact form of visualization which is then implemented in multiplatform application giving inside view to particular steps of decomposition. User of this application can interactively change order of decomposition steps, monitor efficiency or search for interesting relations in input data.*

**Klíčová slova:** booleovská dekompozice matic; vizualizace; formální konceptuální analýza; formální koncept

**Keywords:** boolean matrix decomposition; visualization; formal concept analysis; formal concept

Děkuji panu doc. Janu Outratovi za vedení diplomové práce, zvláště za užitečné rady a podněty při konzultacích.

*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

# 1 Úvod

Žijeme v době, kdy je zvykem sbírání a ukládání dat téměř o čemkoliv. Není tak velké překvapení, že se velké popularitě těší analýza těchto dat. Koneckonců například ve sportu může znalost detailních informací o oponentovi hrát klíčovou roli.

Jedním z formátů dat, které lze zkoumat, jsou takzvaná *objekt-atributová* data. V nich je určitý objekt popsán atributy, které vlastní. Například nechť množina objektů je dána lidmi a množina atributů jsou jimi konzumované pokrmy. Tato *objekt-atributová* data lze jednoduše reprezentovat binární maticí, skutečnost, že člověk $i$ konzumoval pokrm $j$, vyjadřuje hodnota 1 na pozici řádku $i$ a sloupce $j$. Z těchto dat lze vydedukovat určité vzory a závislosti. Na základě konzumovaných pokrmů můžeme rozpoznat vegetariány, lidi s alergií na lepek nebo skupinu lidí, kterým v budoucnu hrozí zdravotní obtíže, protože konzumují nezdravé potraviny.

Tyto skupiny nazýváme faktory a jsou nalezeny použitím takzvané *faktorové analýzy* na vstupní booleovskou matici. Zde je nutné říci, že pojmenování faktorů musí provést odborník, který dané problematice rozumí. V této situaci vyvstává problém s přehledností, protože faktory jsou tvořeny indexy řádků a sloupců v booleovské matici, a počet faktorů je závislý na velikosti a hustotě vstupní matice, tudíž jich bývá zpravidla hodně. Právě i s touto nepřehledností má pomoci aplikace popsaná v této práci, neboť uživateli umožňuje pojmenovat jak objekty, tak atributy vstupních dat. Uživatel potom nepracuje pouze s indexy objektů a atributů, ale s konkrétními jmény, faktory jsou poté jednodušeji pojmenovatelné a celá vstupní data lze snáze analyzovat.

Pokud známe potřebné faktory, lze pomocí nich zkonstruovat dvě nové matice, kde první matice popisuje vztah mezi objekty a faktory, druhá vztah mezi faktory a atributy. Tyto matice jsou výsledkem booleovské dekompozice původní matice, pokud jejich binárním vynásobením dostaneme právě onu původní booleovskou matici.

V práci jsou rozebrány teoretické základy, nutné pro porozumění booleovské dekompozici matic, také je popsána přesná podoba vizualizace jednotlivých faktorů na vstupních datech a výsledná aplikace.

# 2 Booleovská dekompozice matic

Dekompozice booleovské matice $I$ o rozměrech $m \times n$ spočívá v nalezení dvou booleovských matic $A$ a $B$ takových, že po jejich binárním vynásobení dostaneme původní matici $I$. Hledané booleovské matice $A$ a $B$ mají postupně rozměry $m \times k$ a $k \times n$. Krom nalezení příslušných matic je cílem dekompozice také to, aby vnitřní rozměr $k$ hledaných matic byl nejmenší možný. Uplatnění booleovské dekompozice matic lze nalézt například v řešení problémů jako Role mining, Market basket analysis, nebo Topic identification.

## 2.1 Základní pojmy

**Definice 1 (Binární matice)**
Matici $A$ o rozměrech $m \times n$ nazveme binární, pokud pro každé $a_{ij}$ platí $a_{ij} \in \{0, 1\}$, kde $i \in \{1, 2, .., m\}$ a $j \in \{1, 2, ..n\}$.

Z předchozí definice je zřejmé, že v binární matici mohou prvky nabývat pouze hodnot 1, nebo 0, které jsou někdy nazývané také *pravda* a *nepravda*. Níže je uveden příklad binární matice, reprezentující data popsaná tabulkou 1.

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

**Definice 2 (Násobení binárních matic)**
Mějme binární matice $A$ a $B$ postupně o rozměrech $m \times k$ a $k \times n$, skutečnost, že matice $I$ vznikla binárním vynásobením těchto matic, zapíšeme jako $I = A \circ B$, kde operace $\circ$ je definována následovně:

$$(A \circ B)_{ij} = \bigvee_{l=1}^{k} A_{il} \wedge B_{lj}$$

Symboly $\vee$ a $\wedge$ reprezentují postupně logické spojky pro konjunkci a disjunkci. Pro příklad binárního násobení matic viz podkapitolu 2.2.

**Definice 3 (Rozklad binární matice)**
Rozkladem binární matice $I$ velikosti $m \times n$ nazveme dvojici matic $A, B$ postupně o rozměrech $m \times k$ a $k \times n$ takových, že $I = A \circ B$. Číslo $k$ představuje společný rozměr matic $A$ a $B$.

**Definice 4 (Booleovský rank)**
Mějme binární matici $I$, pak nejmenší číslo $k$, pro které existuje rozklad matice $I$ splňující definici 3, nazveme Booleovským rankem této matice. Značíme $rank_B(I)$.

**Problém 1 (Nalezení optimálního rozkladu)**
Nechť je známa binární matice $I$ velikosti $m \times n$, pak hledáme takový rozklad, kde společný rozměr $k$ je nejmenší možný. Tedy $k$ je booleovským rankem $I$.

**Věta 1**
*Problém nalezení optimálního rozkladu pro zadanou binární matici I je NP-těžký. Rozhodovací verze problému, tedy pro zadanou binární matici I a kladné číslo k rozhodni, zda existuje optimální rozklad, je NP-úplný [3] [4].*

*Důkaz*
Ukážeme redukci *Set basis problému* na problém nalezení optimálního rozkladu binární matice. Z vlastností redukce [6], NP-těžkosti *Set basis problému* a NP-úplnosti jeho rozhodovací verze [7] dostaneme požadované tvrzení.

*Set basis problém*(SBP) - rozhodovací verze
Vstup: Konečné univerzum $U = \{1, ..., m\}$, systém množin $S = \{S_1, ..., S_n\}$ kde $S_i \subseteq U$ a kladné číslo $k$.
Výstup: Pravda, pokud existuje systém množin $C = \{C_1, ..., C_k\}$, kde $C_l \subseteq U$ pro $l = 1, ..., k$, takový, že pro každou $S_i$ existuje $D_i \subseteq C$ taková, že $\cup D_i = S_i$, jinak nepravda

Optimalizační verze SBP problému spočívá v nalezení co nejmenšího systému množin $C$, splňujícího stejné podmínky. Mějme instanci problému SBP, pak vstupní matici $I$ velikosti $m \times n$ pro problém nalezení optimálního rozkladu získáme tak, že položíme $I_{ij} = 1$ právě když $j \in S_i$. Pro $C = \{C_1, ...C_k\}$ jako řešení SBP, lze ověřit, že $I = A \circ B$, pokud $A_{il} = 1$ právě když $C_l \in D_i$ a $B_{lj} = 1$ právě když $j \in C_l$. □

Příklad 1 (Redukce SBP na problém rozkladu)
Tento příklad slouží pro ilustraci redukce popsané v předchozím důkazu. Mějme instanci problému SBP, kde:
$U = \{1, 2, 3, 4, 5, 6\}$,
$S = \{\{3, 4, 5, 6\}, \{3, 4\}, \{5, 6\}, \{1, 2\}, \{1, 2, 5, 6\}\}$.
Řešením této instance SBP nechť je množina $C = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$ s $k = 3$. Tedy jednotlivé množiny $D_i$ jsou dány následovně:
$D_1 = \{\{3, 4\}, \{5, 6\}\}$, $D_2 = \{\{3, 4\}\}$, $D_3 = \{\{5, 6\}\}$, $D_4 = \{\{1, 2\}\}$ a
$D_5 = \{\{1, 2\}, \{5, 6\}\}$. Lze snadno vidět, že $\cup D_i = S_i$ pro $i = 1, ..., 5$. Jedná se tedy o správné řešení. Odpovídající matice $I$ a dílčí matice $A, B$ takové, že $I = A \circ B$ jsou dány následovně:

$$I = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}, \; A = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \; B = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

## 2.2 Modelový příklad aplikace

Jak bylo zmíněno výše, booleovskou dekompozici matic lze využít například při řešení takzvaného Role mining problému. Mějme středně velkou firmu používající informační systém. Každý uživatel v tomto systému má přesně nastavená oprávnění, do jakých částí aplikace má přístup a do jakých ne. S nárůstem uživatelů a rozšiřováním informačního systému začne být pro administrátora velmi složité spravovat takováto oprávnění. Rozhodne se, že bude mnohem jednoduší používat řízení přístupu založené na rolích. Tedy uživatel, který měl dříve oprávnění přijímat nové zaměstnance, uzavírat dohody o provedení práce a organizovat pohovory, bude mít nyní roli HR manažer a všechna jeho nynější oprávnění budou obsažena v této roli.

Role mining problém spočívá právě v identifikování těchto rolí, pokud je znám pouze původní seznam uživatelů a jejich jednotlivá oprávnění, například daný tabulkou 1.

| | o1 | o2 | o3 | o4 | o5 | o6 |
|---|---|---|---|---|---|---|
| u1 | | | × | × | × | × |
| u2 | | | × | × | | |
| u3 | | | | | × | × |
| u4 | × | × | | | | |
| u5 | × | × | | | × | × |

Tabulka 1: Uživatelé a jejich oprávnění

Informace obsažené v tabulce 1 lze reprezentovat binární maticí tak, že pokud uživatel má dané oprávnění, umístíme na příslušnou pozici v matici hodnotu 1 jinak 0, matici označme $I$. Výsledek booleovské dekompozice takové matice je popsán následujícím vztahem:

$$I = \begin{array}{c} \\ \text{u1} \\ \text{u2} \\ \text{u3} \\ \text{u4} \\ \text{u5} \end{array} \begin{array}{c} \begin{array}{ccc} \text{HR} & \text{IT} & \text{Pr.} \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array} \circ \begin{array}{c} \begin{array}{cccccc} \text{o1} & \text{o2} & \text{o3} & \text{o4} & \text{o5} & \text{o6} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \begin{array}{c} \\ \text{HR.} \\ \text{IT} \\ \text{Pr.} \end{array}$$

Označme druhou matici $A$ a třetí $B$. Výsledek dekompozice můžeme přečíst následovně, řádky matice $A$ reprezentují uživatele, sloupce reprezentují hledané role, ty lze chápat jako množinu uživatelů s určitou skupinou oprávnění. Hodnota 1 říká, zda uživatel má danou roli, tedy zda patří do skupiny uživatelů s oprávněními náležícími této roli. V matici $B$ řádky reprezentují postupně ty samé role jako sloupce v matici $A$ a sloupce reprezentují oprávnění. Hodnota 1 udává, kdy které roli přísluší jaké oprávnění. Nyní přesně víme, kteří uživatelé mají jaké

role, a známe přesná oprávnění těchto rolí. Administrátor ještě na základě výše popsaných vlastností matic $A$ a $B$ tyto role pojmenuje. V příkladu je pro ilustraci zvoleno postupně HR, IT a Pr. jako zkratky pro Human Resources, IT pracovník a programátor. Administrátor nyní může kontrolovat přístup k daným částem aplikace pouze za pomocí nalezených rolí.

## 2.3 Metody pro klasické matice

Dekompozice matice, kde jednotlivé elementy nabývají reálných hodnot, je hojně studované téma. Za léta výzkumu bylo objeveno mnoho metod pro řešení. Nabízí se tedy již existující metody nebo jejich upravené verze použít i pro řešení dekompozice booleovské matice. Použití těchto metod na booleovské matice ale často vede k reálným hodnotám v dílčích maticích. Tato skutečnost znesnadňuje interpretaci vztahu mezi objekty a faktory, popřípadě faktory a atributy. V této kapitole je shrnut přehled nejznámějších metod pro dekompozici klasických matic a jejich příklady použití na booleovské matice. Při tvorbě této kapitoly bylo čerpáno zejména ze zdroje informací [8]. Pro všechny příklady v této kapitole uvažujme následující matici $I$.

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

### 2.3.1 LU

LU dekompozice rozkládá vstupní matici $A$ velikosti $m \times n$ na dílčí matice $L$ a $U$ postupně o rozměrech $m \times k, k \times n$, kde $k = min(m, n)$ tak, že $A = LU$ za použití klasického násobení matic. Matice $L$ je dolní trojúhelníková matice s jedničkami na diagonále, $U$ je horní trojúhelníková matice vzniklá aplikací Gaussovy eliminační metody na $A$. $U$ je tedy v redukovaném stupňovitém tvaru. Pro LU dekompozici existují dvě varianty používající vhodnou transformaci vstupní matice $A$:

- Varianta s částečným pivotováním – pro transformaci matice $A$ používá pouze permutaci řádků. Předpis dekompozice je pak $PA = LU$, kde matice $P$ tvoří právě permutaci řádků matice $A$.
- Varianta s úplným pivotováním – používá navíc i permutování sloupců tvořené maticí $Q$, předpis pro dekompozici je pak $PAQ = LU$.

LU dekompozice se využívá například pro řešení soustavy rovnic, nalezení inverzní matice či vypočtení determinantu.

Příklad 2 (LU dekompozice)

Pro výše definovanou matici $I$ je výsledek LU dekompozice s částečným pivotováním tvořen maticemi $P, L$ a $U$ danými následovně:

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} U = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Pro výpočet byla použita metoda `lu` z knihovny `scipi.linalg` pro jazyk Python.

Použití LU dekompozice na booleovské matice není vhodné hned z několika důvodů. Společný rozměr $k$ je roven menšímu z čísel $m, n$, tedy není zde žádná šance na zjištění zajímavých analytických dat ze vstupní binární matice. Dále se v maticích $L$ a $U$ mohou vyskytovat reálné hodnoty, pro které není jasné, jak je interpretovat ve vztahu objekt-faktor, popřípadě faktor-atribut.

### 2.3.2 SVD

**Definice 5 (Ortogonální matice)**

Matice $A \in \mathbb{R}^{n \times n}$ je ortogonální, pokud platí $A^T = A^{-1}$

Tedy čtvercová matice je ortogonální, pokud její transponovaná matice je zároveň maticí inverzní. Alternativně lze říci, že matice $A$ je ortogonální, právě když součin s jí příslušnou inverzní maticí je roven jednotkové matici.

**Definice 6 (Norma matice)**

Zobrazení $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ nazýváme normou matice $A$, pokud platí následující podmínky:

1. $f(A) \geq 0$, $A \in \mathbb{R}^{m \times n}$, kde $f(A) = 0$ pk. $A = 0$
2. $f(A + B) \leq f(A) + f(B)$, $A, B \in \mathbb{R}^{m \times n}$
3. $f(\alpha A) = |\alpha| f(A)$, $\alpha \in \mathbb{R}, A \in \mathbb{R}^{m \times n}$

**Definice 7 (Frobeinova norma)**

Frobeinova norma matice $A \in \mathbb{R}^{m \times n}$ je norma daná následujícím předpisem:

$$||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

Metoda *Singular Value Decomposition* (SVD) rozkládá vstupní matici $A$ na součin tří matic $U\Sigma V^T$, kde $U$ je $m \times m$ ortogonální matice, $\Sigma$ je $m \times n$ diagonální matice s nezápornými hodnotami $\sigma_{ii}$ na diagonále, které jsou uspořádané sestupně, a $V$ je $n \times n$ ortogonální matice. Hodnoty $\sigma_{ii}$ jsou singulární hodnoty matice $A$, sloupce $U$ jsou pak levé singulární vektory a sloupce $V$ pravé singulární vektory $A$.

SVG vrací optimální aproximaci vstupní matice $A$ pro rank $k$ vzhledem k Frobeinově normě. Optimální aproximaci pro rank $k$ lze nalézt položením všech singulárních hodnot kromě $k$ nejvyšších na 0.

Příklad 3 (SVD dekompozice)

Uvažujme booleovskou matici $I$ definovanou výše. Dekompozice pomocí SVD pro $k = 3$ je dána následovně:

$$(U\Sigma)_{\_,1:3} = \begin{pmatrix} 1.732 & 1.0 & -0.0 \\ 0.577 & 1.0 & -0.816 \\ 1.155 & 0.0 & 0.816 \\ 0.578 & -1.0 & -0.816 \\ 1.732 & -1.0 & 0.0 \end{pmatrix}$$

$$(V^T)_{1:3,\_} = \begin{pmatrix} 0.289 & 0.289 & 0.289 & 0.289 & 0.577 & 0.577 \\ -0.5 & -0.5 & 0.5 & 0.5 & 0.0 & 0.0 \\ -0.408 & -0.408 & -0.408 & -0.408 & 0.408 & 0.408 \end{pmatrix}$$

$$\sigma_{11} = 2.828,\ \sigma_{22} = 2.0,\ \sigma_{33} = 1.414$$

Kde $(U\Sigma)_{\_,1:3}$ je výběr všech řádků a prvních 3 sloupců z matice $U\Sigma$, $(V^T)_{1:3,\_}$ jsou první tři řádky a všechny sloupce z $V^T$ a $\sigma_{ii}$ pro $i = 1, 2, 3$ představují první tři hodnoty na diagonále matice $\Sigma$, tedy singulární hodnoty matice $I$. Matice $U\Sigma$ má rozměr $5 \times 6$, ale poslední tři sloupce jsou díky vynásobení s $\Sigma$ nulové. Stejně tak $V^T$ má velikost $6 \times 6$, ale při násobení s $U\Sigma$ jsou poslední tři řádky ignorovány právě díky nulovým sloupcům v $U\Sigma$. Počet singulárních hodnot zde tedy určuje společný rozměr pro násobení matic. Pro výše popsané matice platí:

$$(U\Sigma)_{\_,1:3}(V^T)_{1:3,\_} \approx I$$

$$(U\Sigma)_{\_,1:2}(V^T)_{1:2,\_} \approx \begin{pmatrix} -0.0 & -0.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ -0.333 & -0.333 & 0.667 & 0.667 & 0.333 & 0.333 \\ 0.333 & 0.333 & 0.333 & 0.333 & 0.667 & 0.667 \\ 0.667 & 0.667 & -0.333 & -0.333 & 0.333 & 0.333 \\ 1.0 & 1.0 & -0.0 & -0.0 & 1.0 & 1.0 \end{pmatrix}$$

Tedy pro $k = 3$ nalezne SVD přesnou dekompozici $I$, nicméně pro $k = 2$ je to pouze nejpřesnější aproximace $I$. Lze tedy říci, že neexistují matice $A, B$ o rozměrech $5 \times 2$ a $2 \times 6$ takové, že $I = AB$. Pro výpočet byla použita metoda `svd` z knihovny `scipy.linalg` pro jazyk Python.

Z předchozího příkladu lze vidět, že pomocí SVD lze sice zmenšit vnitřní rozměr $k$ při dekompozici, nicméně v dílčích maticích $U\Sigma$ a $V^T$ mohou být stále i záporné hodnoty, u kterých není jasná interpretace ve vztahu objektů a faktorů, popřípadě faktorů a atributů.

### 2.3.3 NMF

*Non neagative Matrix Factorization*(NMF) je metoda pro nalezení rozkladu vstupní matice $V$ na součin matic $WH$, kde matice $V, W, H$ mají postupně rozměry $m \times n$, $m \times k$, $k \times n$ a mohou obsahovat pouze nezáporné hodnoty [9]. Díky nezáporným hodnotám může být každý sloupec $V$ vypočten jako lineární kombinace sloupcových vektorů $W$ za použití koeficientů z $H$, tedy:

$$v_{\_,i} = Wh_{\_,i}$$

kde $v_{\_,i}$ je $i$-tý sloupec $V$ a $h_{\_,i}$ je $i$-tý sloupec $H$. Obecně použití NMF má výhody v redukci vnitřního rozměru $k$ matic $W, H$ a také například oproti SVD se lze vyhnout ortogonálním maticím.

Příklad 4 (NMF dekompozice)
NMF dekompozice výše definované matice $I$ je dána následovně:

$$W = \begin{pmatrix} 0.871 & 0.0 & 0.892 \\ 0.0 & 0.0 & 0.892 \\ 0.871 & 0.0 & 0.0 \\ 0.0 & 0.98 & 0.0 \\ 0.871 & 0.98 & 0.0 \end{pmatrix}$$

$$H = \begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 1.149 & 1.149 \\ 1.021 & 1.021 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.121 & 1.121 & 0.0 & 0.0 \end{pmatrix}$$

Potom platí $I \approx WH$ za použití běžného násobení matic nebo násobení definovaného pomocí lineárních kombinací výše. Vnitřní rozměr matic $W, H$ je roven 3, tedy nejmenšímu možnému, viz příklad a optimalita SVD. Pro použití binárního násobení by bylo třeba zaokrouhlit hodnoty v dílčích maticích, nicméně pak nemusí platit, že výsledkem binárního vynásobení $W$ a $H$ je přesně $I$. Pro výpočet byla použita třída `NMF` z knihovny `sklearn.decomposition` pro jazyk Python.

**Shrnutí obecných metod**
Pro nalezení dekompozice matic s reálnými hodnotami existuje mnoho metod, kromě výše popsaných také například *principal component analysis* (PCA) nebo *independent compomenet analysis* (IDA). K aplikaci klasických metod na booleovské matice bylo navrženo několik rozšíření. Pro PCA viz [10], [11]. Většina

klasických metod, i jejich rozšíření pro booleovské matice ale vede na reálné a často i záporné hodnoty v dílčích maticích. Pro dekompozici booleovské matice je ale pochopitelné chtít, aby i dílčí matice nabývaly pouze binárních hodnot. V takovém případě lze použít binární násobení matic a zároveň dílčí matice poskytují zajímavý, původně skrytý pohled na vstupní data. Způsob, jak nalézt čistě booleovskou dekompozici, popisuje Formální konceptuální analýza.

# 3 Formální konceptuální analýza (FKA)

Formální konceptuální analýza je metoda zabývající se analýzou tabulkových dat, která popisují vztahy mezi objekty a atributy. Její hlavní výhodou je, že poskytuje jiný pohled na data a mnohdy dokáže odhalit ve vstupních datech informace, které nejsou na první pohled zřejmé. Aplikace této metody lze nalézt v různých odvětvích, jako je například *data-minig* [1] nebo *machine learning* [2]. Hlavním pojmem formální konceptuální analýzy je formální koncept, který můžeme chápat jako shluk v datech. Formální koncepty spolu s relací částečného uspořádání poté tvoří konceptuální svaz. V této kapitole jsou popsány pojmy potřebné k porozumění základům formální konceptuální analýzy a také její použití při booleovské dekompozici matic. Při zpracovávání toho tématu bylo čerpáno ze zdrojů informací [1] a [2].

**FKA a booleovská dekompozice matic**
Mějme binární matici $I$ velikosti $m \times n$, popisující vztah mezi objekty a atributy. Konkrétně řekneme, že objekt $i$ má atribut $j$, pokud $I_{ij} = 1$. Booleovskou dekompozici $I = A \circ B$ kde $A$ má velikost $m \times k$ a $B$ $k \times n$ lze z pohledu FKA chápat jako nalezení $k$ faktorů popisující data v $I$. Faktor zde můžeme interpretovat jako skupinu objektů se specifickými atributy. Pokud $A_{il} = 1$, pak řekneme, že objekt $i$ patří do faktoru (skupiny objektů) $l$, podobně pokud $B_{lj} = 1$ řekneme, že faktor $l$ má atribut $j$, tedy $j$ je jedním z charakteristických atributů faktoru $l$. Ekvivalentně, ve vstupní matici $I$ má objekt $i$ atribut $j$, pokud existuje faktor $l$ takový, že $i$ patří do $l$ a $j$ je charakteristickým atributem $l$.

## 3.1 Základní pojmy

**Definice 8 (Formální kontext)**
Formální kontext je trojice $\langle X, Y, I \rangle$, kde $X$ chápeme jako množinu objektů, $Y$ jako množinu atributů a $I$ je binární relace popisující, kdy objekt $x \in X$ má atribut $y \in Y$.

Uvažujme postupně množinu objektů $X = \{u1, u2, u3, u4, u5\}$, množinu atributů $Y = \{o1, o2, o3, o4, o5, o6\}$ a binární relaci $I = \{\langle u1, o3\rangle, \langle u1, o4\rangle, \langle u1, o5\rangle, \langle u1, o6\rangle, \langle u2, o3\rangle, \langle u2, o4\rangle, \langle u3, o5\rangle, \langle u3, o6\rangle, \langle u4, o1\rangle, \langle u4, o2\rangle, \langle u5, o1\rangle, \langle u5, o2\rangle, \langle u5, o5\rangle, \langle u5, o6\rangle\}$. Takto popsaný formální kontext odpovídá tabulce uživatelů a jejich oprávnění v kapitole 2.2, pro připomenutí zmíněna níže, odpovídající binární matice je pak uvedena v 2.3.

[1] získávaní skrytých informací z dat
[2] umožnění počítači učit se

| | o1 | o2 | o3 | o4 | o5 | o6 |
|---|---|---|---|---|---|---|
| u1 | | | × | × | × | × |
| u2 | | | × | × | | |
| u3 | | | | | × | × |
| u4 | × | × | | | | |
| u5 | × | × | | | × | × |

Uživatelé a jejich oprávnění

**Definice 9 (Šipkový operátor ↑)**
Šipkový operátor ↑ přiřazuje skupině objektů jejich společné atributy. Je tedy definován jako zobrazení $\uparrow : \ 2^X \to 2^Y$ dáno následujícím předpisem:

$$A^{\uparrow} = \{y \in Y \mid \text{pro } \forall x \in A : \langle x, y\rangle \in I\}, \text{ kde } A \subseteq X$$

Pokud budeme uvažovat formální kontext popsaný tabulkou 1,
pak $\{u1, u2\}^{\uparrow} = \{o3, o4\}$. Jinými slovy, $o3$ a $o4$ jsou jediné atributy společné pro objekty $u1$ a $u2$.

**Definice 10 (Šipkový operátor ↓)**
Šipkový operátor ↓ přiřazuje skupině atributů takové objekty, které je sdílí. Tedy je definován jako zobrazení $\downarrow : \ 2^Y \to 2^X$ dáno následující předpisem:

$$B^{\downarrow} = \{x \in X \mid \text{pro } \forall y \in B : \langle x, y\rangle \in I\}, \text{ kde } B \subseteq Y$$

Opět na formálním kontextu popsaným tabulkou 1 lze ukázat,
že $\{o5, o6\}^{\downarrow} = \{u1, u3, u5\}$. Tedy $u1, u3$ a $u5$ jsou jediné objekty sdílející atributy $o5$ a $o6$.

**Definice 11 (Formální koncept)**
Mějme formální kontext $\langle X, Y, I\rangle$, pak formální koncept je každá dvojice $\langle A, B\rangle$ splňující $A^{\uparrow} = B$ a $B^{\downarrow} = A$ kde $A \subseteq X, B \subseteq Y$. Množinu všech formálních konceptů pro $\langle X, Y, I\rangle$ značíme $\mathcal{B}(X, Y, I)$, tedy:

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

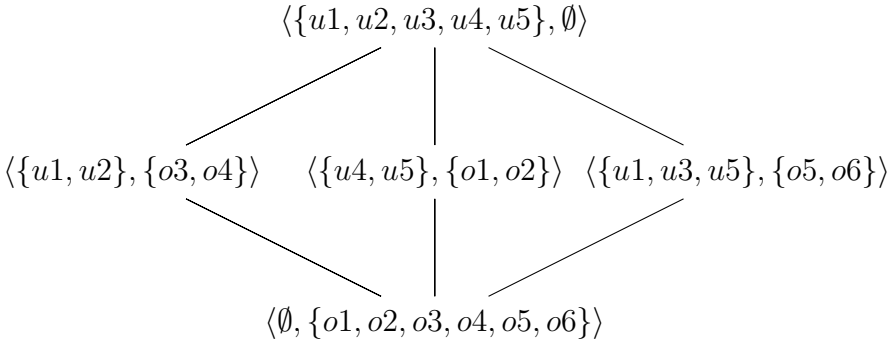
Formální koncept se skládá z množiny objektů $A$ (extent formálního konceptu) a množiny atributů $B$ (intent formálního konceptu), kde právě atributy z $B$ jsou společné všem objektům z $A$ a zároveň objekty z $A$ jsou sdílené všemi atributy z $B$. Těmto formálním konceptům se také někdy říká faktory. Formální kontext daný tabulkou 1 obsahuje tři netriviální formální koncepty, $\langle\{u1, u2\}, \{o3, o4\}\rangle$, $\langle\{u1, u3, u5\}, \{o5, o6\}\rangle$ a $\langle\{u4, u5\}, \{o1, o2\}\rangle$.

**Definice 12 (Konceptuální svaz)**

Konceptuální svaz je tvořen množinou všech formálních konceptů $\mathcal{B}(X, Y, I)$ a binární relací částečného uspořádání $\leq$, definovanou na této množině následovně:

$$\langle A_1, B_1 \rangle \leq \langle A_2, B_2 \rangle \text{ právě když } A_1 \subseteq A_2 \text{ (nebo ekvivalentně } B_2 \subseteq B_1)$$

Konceptuální svaz je úplný svaz, tedy pro každé $K \subseteq \mathcal{B}(X, Y, I)$ existuje supremum a infimum [1]. Pomocí výše popsané relace částečného uspořádání lze porovnávat formální koncepty. Můžeme říci, že jeden formální koncept je obecnější než jiný, nebo naopak konkrétnější. Nutno podotknout, že formální koncepty mohou být také nesrovnatelné, to se stane právě tehdy, když $A_1 \cap A_2 = \emptyset$ (nebo ekvivalentně $B_1 \cap B_2 = \emptyset$). Takto částečně uspořádanou množinu lze znázornit Hasseovým diagramem.




Obrázek 1: Hasseův diagram konceptuálního svazu.

Na obrázku 1 je ilustrován konceptuální svaz odpovídající formálnímu kontextu danému tabulkou 1. Zde je například vidět, že triviální formální koncept $\langle \{u1, u2, u3, u4, u5\}, \emptyset \rangle$, který můžeme pojmenovat jako "uživatelé systému", je obecnější než formální koncept $\langle \{u1, u2, u3\}, \{o5, o6\} \rangle$, který reprezentuje skupinu programátorů. Pro výpočet množiny $\mathcal{B}(X, Y, I)$ a případné generování konceptuálního svazu je známo několik variant algoritmů, viz například [12] a [5].

## 3.2 Formální koncepty jako optimální faktory

V této kapitole je popsáno, jak využít formální koncepty nalezené ve vstupní matici $I$ velikosti $m \times n$, pro konstrukci matic $A, B$ o velikostech $m \times k$ a $k \times n$ takových, že $I = A \circ B$. Zároveň je ukázáno, že právě za použití formálních konceptů pro konstrukci matic $A, B$ lze minimalizovat jejich vnitřní rozměr $k$ [3].

**Definice 13 (Faktorové matice)**

Mějme formální kontext $\langle X, Y, I \rangle$ a jemu příslušnou množinu všech formálních konceptů $\mathcal{B}(X, Y, I)$. Nechť $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$, tj. $\mathcal{F} = \{\langle A_1, B_1 \rangle, ..., \langle A_k, B_k \rangle\}$.

Označme postupně matici $A_{\mathcal{F}}$ jako objekt-faktorovou a $B_{\mathcal{F}}$ jako faktor-atributovou matici. Samotné matice jsou dány následujícím předpisem:

$$(A_{\mathcal{F}})_{il} = \begin{cases} 1, & \text{pokud } i \in A_l \\ 0, & \text{pokud } i \notin A_l \end{cases} \quad (B_{\mathcal{F}})_{lj} = \begin{cases} 1, & \text{pokud } j \in B_l \\ 0, & \text{pokud } j \notin B_l \end{cases} \quad \text{pro } l = 1, ..., k$$

Pak řekneme, že $l$-tý sloupec $A_{\mathcal{F}}$ tj. $(A_{\mathcal{F}})_{\_,l}$ je charakteristický vektor extentu $A_l$ a $(B_{\mathcal{F}})_{l,\_}$ je charakteristický vektor intentu $B_l$.

**Věta 2 (Univerzálnost formálních konceptů)**
*Pro každou binární matici $I$ existuje $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ taková, že $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$.*

*Důkaz*
Předpokládejme, že $I_{ij} = 1$, pak existuje formální koncept $\langle C, D \rangle \in \mathcal{B}(X, Y, I)$ takový, že $i \in C$ a $j \in D$. Nechť $\mathcal{F} = \mathcal{B}(X, Y, I)$, pak dostaneme $(A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij} = 1$ právě když existuje $l$ takové, že $(A_{\mathcal{F}})_{il} = 1$ a $(B_{\mathcal{F}})_{lj} = 1$. Takové $l$ existuje, právě když existuje formální koncept $\langle C_l, D_l \rangle \in \mathcal{B}(X, Y, I)$ s $i \in C_l$ a $j \in D_l$. Takový formální koncept existuje, právě když $I_{ij} = 1$, což je předpoklad. □

Předchozí věta říká, že pro každou binární matici lze nalézt její booleovskou dekompozici vypočtením množiny všech formálních konceptů a sestrojením faktorových matic.

**Věta 3 (Optimalita formálních konceptů)**
*Nechť $I = A \circ B$ pro binární matice $A, B$ o rozměrech $m \times k$ a $k \times n$, pak existuje $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ taková, že $|\mathcal{F}| \leq k$ a zároveň pro odpovídající faktorové matice $A_{\mathcal{F}}, B_{\mathcal{F}}$ o rozměrech $m \times |\mathcal{F}|$ a $|\mathcal{F}| \times n$ platí $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$.*

*Důkaz*
Kostra důkazu je zde shrnuta do 3 bodů, pro podrobný popis viz [3].

1. Formální koncepty jsou maximální obdélníky ve vstupní matici $I$. Tedy do žádného formálního konceptu nelze přidat žádný objekt ani atribut tak, aby se celková velikost (počet objektů × počet atributů) zvětšila a stále platily podmínky definice formálního konceptu. Tyto obdélníky jsou plné jedniček.

2. Mějme $I = A \circ B$ pro matice $A$ velikosti $m \times n$ a $B$ velikosti $n \times k$. Pak $I$ může být vyjádřena jako $\vee$-superpozice (sjednocení) $k$ obdélníků, skládajících se ze samých jedniček. Tyto obdélníky nemusí být nutně největší. Uvažme obdélníky $J_l = A_{\_l} \circ B_{l\_}$, tedy $l$-tý obdélník vznikne vynásobením $l$-tého řádku $A$ a $l$-tého sloupce $B$. Pak platí:

$$I = \bigvee_{l=1}^{k} J_l$$

3. Mějme $I = A \circ B$ pro matice $A$ velikosti $m \times n$ a $B$ velikosti $n \times k$, uvažujme příslušné obdélníky $J_1, ...J_k$. Tyto obdélníky nejsou nutně maximální, ale každý obdélník $J_l$ je obsažen v některém z maximálních obdélníků $J_l^{'}$, který odpovídá formálnímu konceptu. Následná $\vee$-superpozice maximálních obdélníků $J_l^{'}$ má za výsledek původní matici $I$, tj. $\mathcal{F} = \{\langle C_1, D_1\rangle, ..., \langle C_k, D_k\rangle\}$. Jak bylo výše zmíněno, tak více obdélníků, které nejsou maximální, může být obsaženo v jednom maximálním obdélníku, tedy dostáváme $|\mathcal{F}| < k$.

□

Nyní víme, že pro každou binární matici $I$ lze za použití správné množiny $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ sestrojit matice $A_\mathcal{F}, B_\mathcal{F}$ s minimálním možným vnitřním rozměrem $k$ takové, že $I = A_\mathcal{F} \circ B_\mathcal{F}$. Jak sestrojit $\mathcal{F}$ je popsáno dále v textu.

**Definice 14 (Objektové a atributové formální koncepty)**
Mějme formální kontext $\langle X, Y, I\rangle$, pak definujme dvě skupiny formálních konceptů, objektové koncepty $\mathcal{O}(X, Y, I)$ a atributové koncepty $\mathcal{A}(X, Y, I)$ dány následovně:

$$\mathcal{O}(X, Y, I) = \{\langle \{x\}^{\uparrow\downarrow}, \{x\}^{\uparrow}\rangle \mid x \in X\}$$
$$\mathcal{A}(X, Y, I) = \{\langle \{y\}^{\downarrow}, \{y\}^{\downarrow\uparrow}\rangle \mid y \in Y\}$$

**Věta 4 (Povinné formální koncepty)**
*Mějme $I = A_\mathcal{F} \circ B_\mathcal{F}$ pro nějaké $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$,*
*pak $\mathcal{O}(X, Y, I) \cap \mathcal{A}(X, Y, I) \subseteq \mathcal{F}$.*

*Důkaz*
Předpokládejme formální koncept $\langle C, D\rangle \in \mathcal{O}(X, Y, I) \cap \mathcal{A}(X, Y, I)$, tj. $\langle C, D\rangle = \langle \{x\}^{\uparrow\downarrow}, \{x\}^{\uparrow}\rangle$ pro nějaký objekt $x \in X$ a $\langle C, D\rangle = \langle \{y\}^{\downarrow}, \{y\}^{\downarrow\uparrow}\rangle$ pro nějaký atribut $y \in Y$. Pak formální koncept $\langle C, D\rangle$ je jediný z $\mathcal{B}(X, Y, I)$ pokrývající jedničku v matici $I$, na pozici odpovídající objektu $x$ a atributu $y$. Uvažujme jiný formální koncept $\langle C_1, D_1\rangle$, pokrývající tutéž jedničku v matici $I$, z vlastností šipkových operátorů lze ukázat, že v takovém případě $\langle C, D\rangle = \langle C_1, D_1\rangle$. □

**Definice 15 (Faktorové koncepty)**
Množinu $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ nazýváme faktorové koncepty, pokud $I = A_\mathcal{F} \circ B_\mathcal{F}$.

**Nalezení faktorových konceptů**
Pomocí výše popsaných znalostí lze zavést algoritmus, jak pro binární matici $I$ nalézt množinu $\mathcal{F}$ faktorových konceptů. Tento algoritmus lze popsat ve třech krocích:

1. Vypočítej množinu všech formálních konceptů $\mathcal{B}(X, Y, I)$, například pomocí algoritmů v [12].

2. Do $\mathcal{F}$ zařad všechny povinné formální koncepty, tj. takové $\langle C, D\rangle \in \mathcal{O}(X, Y, I) \cap \mathcal{A}(X, Y, I)$.

3. Dokud nejsou pokryté všechny jedničky v $I$, postupně přidávej do $\mathcal{F}$ takové formální koncepty, které pokrývají nejvyšší počet dosud nepokrytých jedniček.

Takto popsaný postup vede na greedy algoritmus. Největší nevýhodou takového algoritmu je, že potřebuje znát množinu všech formálních konceptů. Algoritmus spolu s dalším, efektivnějším, který nepotřebuje znát všechny formální koncepty, je podrobně popsaný v [3].

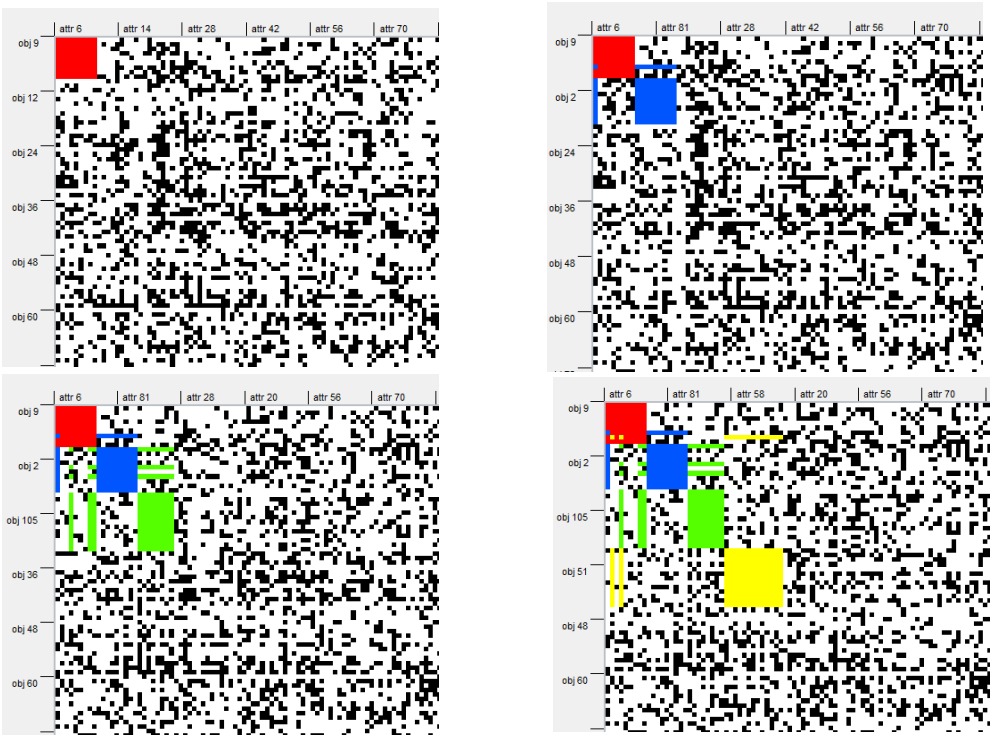
# 4 Vizualizace booleovské dekompozice

Pro binární matici $I$ o velikosti $m \times n$ lze za výsledek booleovské dekompozice považovat množinu $k$ faktorových konceptů $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$. Pomocí množiny faktorových konceptů lze zkonstruovat matice $A_\mathcal{F}, B_\mathcal{F}$ o velikostech $m \times k$ a $k \times n$, pro které platí $I = A_\mathcal{F} \circ B_\mathcal{F}$. Pro účely vizualizace je v níže popsané aplikaci použita právě množina faktorových konceptů. Formální koncept navíc představuje skupinu objektů sdílející stejné atributy, tedy má určitou analytickou hodnotu.

## 4.1 Popis vizualizace

Proces vizualizace booleovské dekompozice binární matice $I$ bere na vstup množinu faktorových konceptů $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$, vypočtenou externím algoritmem. Nechť tedy $\mathcal{F} = \{\langle A_1, B_1\rangle, ..., \langle A_k, B_k\rangle\}$. Pro potřeby vizualizace uvažujme množinu $\mathcal{F}$ s možností uspořádání prvků, tedy seznam neobsahující duplicity. Iniciální pořadí faktorových konceptů je dáno výstupem externího algoritmu, který vypočte $\mathcal{F}$.

Uživatel si v aplikaci může vybrat, které formální koncepty chce vizualizovat, vybraný seznam konceptů označme $\mathcal{M}$. Pro $\mathcal{M}$ platí, že neobsahuje duplicity a zároveň $\mathcal{M} \subseteq \mathcal{F}$. Jednotlivé formální koncepty jsou při vizualizaci postupně slévány k levému hornímu rohu vstupní matice. Dochází tedy k transformaci vstupní matice pomocí záměn řádků a sloupců. První vizualizovaný formální koncept se vždy nachází v levém horním rohu vstupní matice, pravý dolní roh posledního vizualizovaného formálního konceptu určuje část matice, která nesmí být dále transformována. Při vizualizaci dalšího formálního konceptu bude slita jenom ta část, která se nenachází v již neměnné části matice. Obrázek 2 zobrazuje slití prvních čtyř formálních konceptů náhodně generované matice $I$. Přesný postup vizualizace uživatelem vybraných formálních konceptů $\mathcal{M}$ je popsán algoritmem 1.




Obrázek 2: Vizualizace prvních čtyř faktorových konceptů

## 4.2 Algoritmus slévání

Hlavní motivací pro slévání konceptů popsané algoritmem 1 je zobrazit uživateli formální koncepty jako skupiny objektů s příslušnými atributy co možná nejblíže u sebe (z grafického hlediska). Pro uživatele je pak snazší všimnout si například objektů zahrnutých ve více formálních konceptech. S rostoucím množstvím vizualizovaných formálních konceptů nicméně dochází ke zmenšování slévané části, proto je uživateli dána možnost výběru formálních konceptů pro vizualizaci, aby si mohl přehledně zobrazit pouze určité formální koncepty, viz sekci 5.1.

Algoritmus bere na vstup binární matici $I$ a seznam uživatelem vybraných formálních konceptů pro vizualizaci $\mathcal{M}$. Poté dochází k postupnému slévání formálních konceptů z $\mathcal{M}$. Algoritmus dále využívá následující metody:

- `ReplaceKeepSortedExt(list concepts, int e1, int e2)`

  1. Pro všechny formální koncepty pokrývající řádek `e1`, nahradí v extentu (tvořeném indexy řádků) výskyt `e1` prvkem `e2` a zachová prvky extentu setřízené vzestupně.
  2. Pro všechny formální koncepty pokrývající řádek `e2`, nahradí v extentu výskyt `e2` prvkem `e1` a zachová prvky extentu setřízené vzestupně.

3. Aktualizuje informace o formálních konceptech pokrývajících řádky `e1` a `e2`.

- `SwapRow(matrix[][], int r1, int r2)`
  V matici `matix` vymění řádek určený indexem `r1` s řádkem určeným indexem `r2`.

- `ReplaceKeepSortedInt(list concepts, int i1, int i1)`
  Ekvivalentně s `ReplaceKeepSortedExt`, akorát pracuje s intenty, tedy indexy sloupců.

- `SwapCol(matrix[][], int c1, int c2)`
  Ekvivalentně s `SwapRow`, akorát pracuje se sloupci.

**Algoritmus 1** Vizualizace formálních konceptů sléváním

VSTUP: binární matice $I, \mathcal{M} = \{\langle C_1, D_1\rangle, ..., \langle C_l, D_l\rangle\}$

$lastRow \leftarrow 0$
$lastCol \leftarrow 0$
**for** $\langle C_j, D_j\rangle \in \mathcal{M}$ **do**
    **for** $extIdx \leftarrow 0$ **to** $Size(C_j)$ **do**
        $e \leftarrow (C_j)_{extIdx}$
        **if** $lastRow < e$ **then**
            $ReplaceKeepSortedExt(\mathcal{M}, e, lastRow)$
            $SwapRow(I, e, lastRow)$
            $lastRow \leftarrow lastRow + 1$
        **end if**
    **end for**
    **for** $intIdx \leftarrow 0$ **to** $Size(D_j)$ **do**
        $i \leftarrow (D_j)_{intIdx}$
        **if** $lastCol < i$ **then**
            $ReplaceKeepSortedInt(\mathcal{M}, i, lastCol)$
            $SwapCol(I, i, lastCol)$
            $lastCol \leftarrow lastCol + 1$
        **end if**
    **end for**
**end for**

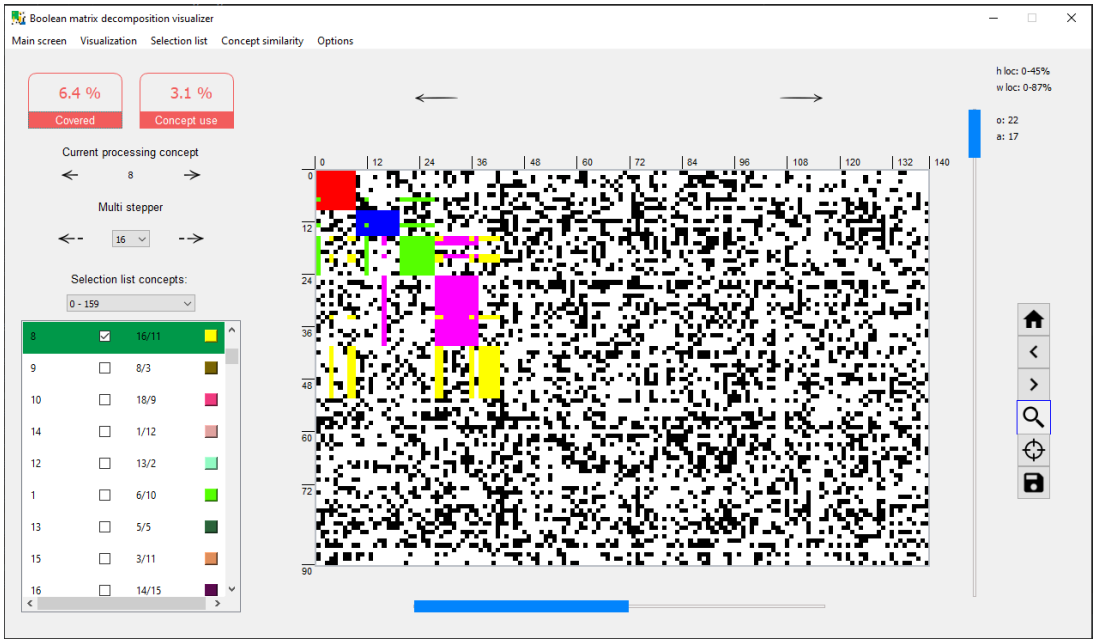
# 5 Aplikace

Vizualizace popsaná v kapitole 4 je spolu s dalšími nástroji pro zkoumání booleovské dekompozice matice $I$ implementována do multiplatformní aplikace. Pro implementaci byl zvolen programovací jazyk Python, jedním z hlavních důvodů volby tohoto jazyka je jeho přehlednost a množství knihoven, které lze při implementaci vizualizace využít. Mezi nejvyužívanější knihovny patří zejména `numpy` pro práci s maticemi a `matplotlib` pro vizualizaci dat. Veškeré použité knihovny a technologie jsou pak popsány v kapitole 8.

V této kapitole je implementovaná aplikace postupně rozdělena do jednotlivých obrazovek, kde ke každé obrazovce je podrobně uvedeno, co na ní uživatel nalezne, a jednotlivé akce, které lze na dané obrazovce provádět. Na závěr jsou uvedeny implementační detaily některých problémů, které bylo při implementování aplikace nutné vyřešit. Popis vstupu a výstupu aplikace je pak uveden v kapitole 6.

## 5.1 Hlavní obrazovka

Hlavní obrazovka aplikace zobrazuje uživateli základní data, jako je například pohled na matici nebo její část, a dává uživateli možnost navigace po částech matice. Dále zobrazuje průběžné informace o počtu pokrytých jedniček a využití formálních konceptů. V neposlední řadě umožňuje uživateli vybrat formální koncepty, které chce vizualizovat.




Obrázek 3: Hlavní obrazovka

Na hlavní obrazovce, ilustrované obrázkem 3, se postupně nachází:

1. **Aktuální podoba vstupní matice** $I$ – představuje největší část hlavní obrazovky, černá barva na pozici odpovídající objektu $i$ a atributu $j$ značí, že objekt $i$ má atribut $j$, tedy v původní matici $I_{ij} = 1$. Černá barva může být při procesu vizualizace nahrazena barvou formálního konceptu. Barvy formálních konceptů jsou na začátku generovány náhodně, pokud nejsou importovány, viz 6. Uživatel si poté může barvy přizpůsobit v aplikaci. Aktuální podoba vstupní matice je vyobrazena ve formě výřezu. Po tomto výřezu se lze interaktivně pohybovat použitím kolečka myši, dostupných scroll-barů a šipek. Pomocí menu na pravé straně může uživatel i různě přibližovat, či oddalovat aktuálně vykreslený výřez matice. Po kliknutí pravým tlačítkem na pozici, kde $I_{ij} = 1$, lze přejít na obrazovku detail souřadnic, detail objektu nebo detail atributu, viz postupně podkapitoly 5.3 a 5.8. Pomocí pravého tlačítka lze také označit koncepty pokrývající danou souřadnici v seznamu konceptů pro vizualizaci, viz bod 2.

2. **Výběr formálních konceptů pro vizualizaci** – nachází se v levém dolním rohu aplikace a určuje, které formální koncepty budou postupně vizualizovány. Kromě výběru formálních konceptů dává uživateli také možnost měnit pořadí zpracování. Toto pořadí může být změněno například přetáhnutím prvku představující formální koncept na jinou pozici pomocí Drag&Drop[3] nebo pravým tlačítkem a příslušnou volbou v menu. Každý prvek v seznamu představuje formální koncept a postupně obsahuje:

   - Jednoznačný identifikátor formálního konceptu.
   - Zaškrtávací pole, určující výběr daného konceptu pro vizualizaci.
   - Informaci o počtu objektů a atributů formálního konceptu.
   - Informaci o barvě formálního konceptu, kterou je poté vykreslen. Po kliknutí na miniaturu barvy může být změněna, nebo celý seznam barev může být importován, viz kapitola 6.

   Aktuálně zpracovaný koncept je v seznamu zobrazen zelenou barvou. Pravým kliknutím na formální koncept v seznamu lze definovat jeho jméno, nebo přejít na obrazovku detail konceptu, viz 5.4. Z důvodu možného velkého množství formálních konceptů lze v tomto seznamu zobrazit pouze určitý počet. Zobrazované formální koncepty může uživatel vybrat v nabídce přímo nad seznamem.

3. **Ovládání průběhu a informace o něm** – je umístěno v levé horní části, zobrazuje uživateli informace o počtu pokrytých jedniček v matici $I$ a také o počtu již vizualizovaných formálních konceptů, oba tyto údaje jsou uvedeny v procentech. Po kliknutí na tlačítko pod procentuální hodnotou je uživatel přesměrován na obrazovku detail průběhu, viz 5.6. Pod těmito údaji se nachází ovládání průběhu vizualizace, uživatel zde může jednotlivě

[3]Technika sloužící k přemisťování objektů na obrazovce pomocí tahu myši.

krokovat postupné vykreslování konceptů. Kliknutím na tlačítko s šipkou doprava je formální koncept zvýrazněn, po druhém kliknutí je provedeno slití algoritmem 1. V části *multi stepper* lze vizualizovat vybraný počet formálních konceptů najednou, tato akce může trvat delší dobu, zvláště pokud je vybráno velké množství konceptů.

4. **Menu pro navigaci po výřezu matice** – nachází se na pravé straně hlavní obrazovky, nabízí uživateli následující možnosti:

   - Návrat na iniciální pozici výřezu matice, tato akce přeruší řetěz historie, viz dále.
   - Posun v historii zpět.
   - Posun v historii dopředu.
   - Přiblížení určité části výřezu. Pro přiblížení je třeba kliknout na tlačítko s lupou a poté tahem myši označit část výřezu matice, kterou chce uživatel přiblížit. Přiblížení tvoří řetěz historie.
   - Zobrazení uživatelem zadané části matice. Tato akce tvoří řetěz historie.
   - Uložení aktuálního výřezu matice do souboru formátu *jpg*.

   V části nad menu pro navigaci se nachází informace o aktuálním výřezu vzhledem k celé matici. Konkrétně *h loc* určuje horizontální pozici aktuálního výřezu vzhledem k počtu sloupců původní matice, *v loc* určuje vertikální pozici vzhledem k počtu řádků. Dále se nad menu nachází informace o aktuálním objektu a atributu vzhledem k pozici, na kterou uživatel kliknul ve výřezu matice.

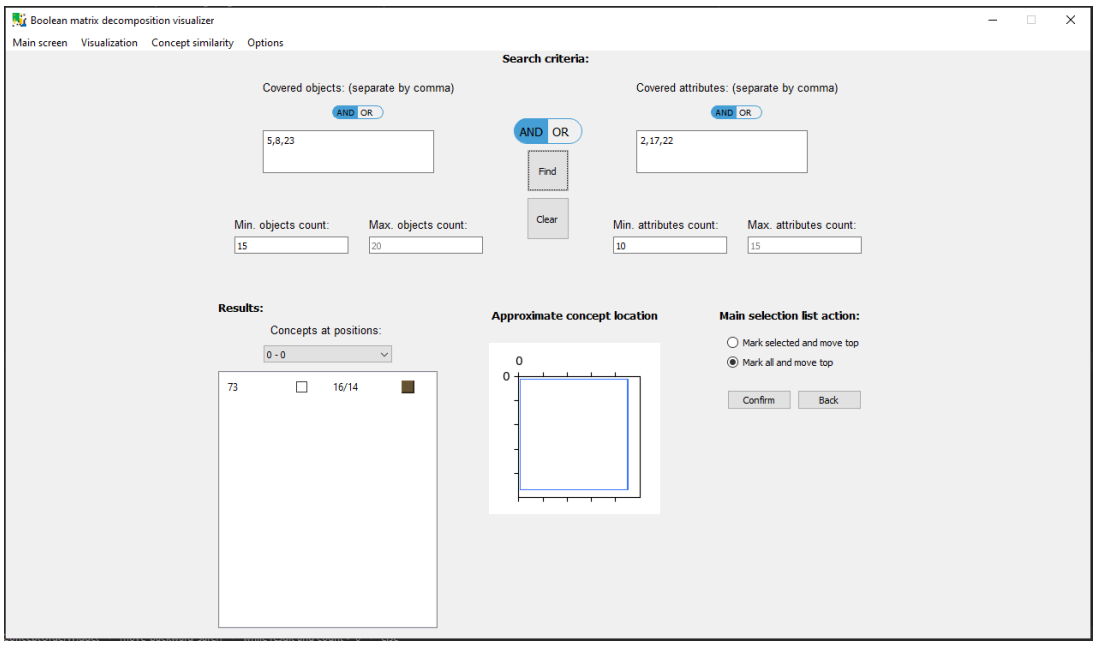
5. **Hlavní menu aplikace** - je umístěno v horním panelu a postupně nabízí uživateli následující možnosti:

   - Návrat na hlavní obrazovku.
   - Restartování vizualizace.
   - Vyhledání formálních konceptů na základě uživatelem zadaných kritérií, nalezené koncepty lze poté označit pro vizualizaci, viz obrazovka 5.2.
   - Odznačit formální koncepty, které ještě nejsou vizualizované, ale jsou označené.
   - Vyhledat konkrétní formální koncept v seznamu pro vizualizaci. Vyhledání je provedeno na základě jednoznačného identifikátoru.
   - Přejít na obrazovku podobnost formálních konceptů. Lze dvěma způsoby, buď pro označené koncepty, nebo bez specifického výběru. Více informací v sekci 5.5.
   - Přejít do nastavení aplikace, viz kapitolu 5.9.

- Spravovat jména jednotlivých objektů a atributů, viz sekci 5.7
- Exportovat aktuální stav aplikace, více viz kapitolu 6.
- Uložení snímku celé aplikace do souboru formátu *jpg*.

## 5.2 Obrazovka výběru formálních konceptů

Dává uživateli pokročilé možnosti, jak vybrat formální koncepty pro vizualizaci nebo obecně hledat formální koncepty na základě uživatelem zadaných vlastností. Nalezené formální koncepty, nebo jejich část může poté uživatel označit pro vizualizaci v seznamu na hlavní obrazovce. Vždy se prohledávají pouze formální koncepty, které ještě nejsou vizualizované. Obrazovka je ilustrovaná obrázkem 4.




Obrázek 4: Obrazovka výběr konceptu

Horní polovina vyhledávací obrazovky se skládá ze dvou částí, v levé části může uživatel specifikovat objekty, v pravé naopak atributy, které mají hledané formální koncepty obsahovat. Uživatel může specifikovat maximální počet objektů či atributů u hledaných formálních konceptů. Lze také zvolit logickou spojku, která bude při vyhledávání použita, na výběr jsou spojky *and* a *or*. Tyto spojky se postupně aplikují na:

- Seznam objektů – při použití logické spojky *and* budou vyhledány formální koncepty takové, které obsahují v extentu všechny uvedené objekty. Při použití logické spojky *or* zase takové, které v extentu obsahují alespoň jeden z uvedených objektů. Množinu formálních konceptů nalezených na základě kritérií týkajících se objektů označme $A$.

- Seznam atributů – ekvivalentně seznamu objektů, pouze se vyhledává na základě atributů. Množinu formálních konceptů nalezenou na základě atributových kritérií označme $B$.
- Nalezené množiny formálních konceptů $A, B$ – pokud jako hlavní logická spojka uprostřed vyhledávácí části je zvolen *and* a objektová i atributová kritéria jsou neprázdná, pak nalezené formální koncepty jsou získány jako $A \cap B$, při použití *or* jako $A \cup B$. Jestliže jsou objektová, nebo atributová kritéria nevyplněna, pak formální koncepty odpovídající nevyplněné části jsou ignorovány.

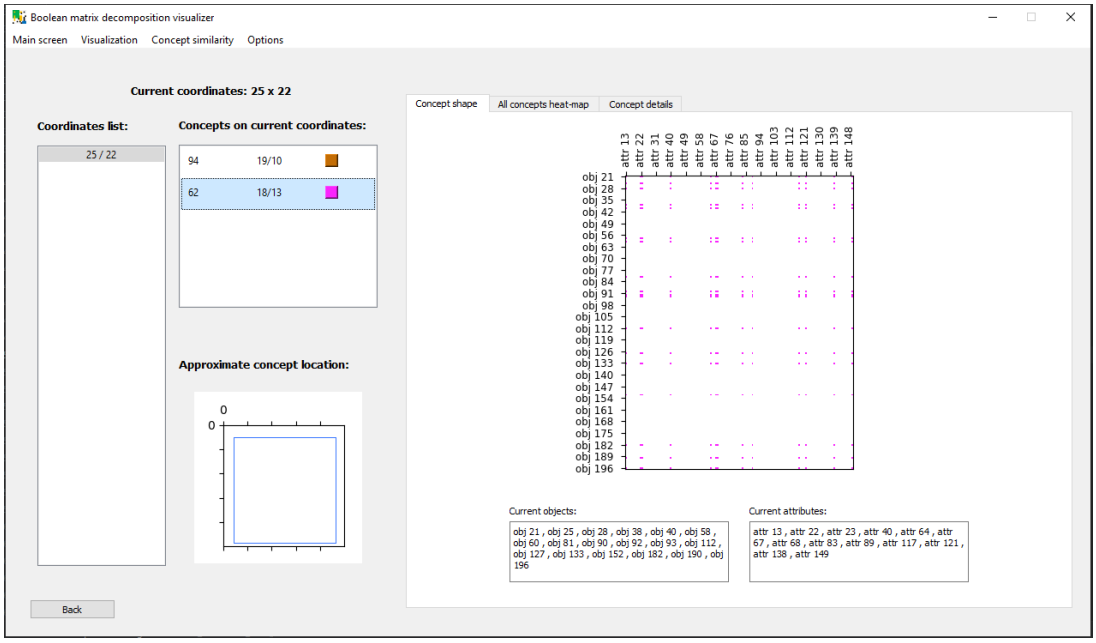
Způsob vyhledávání formálních konceptů na základě uživatelem definovaných kritérií přímo závisí na tom, zda uživatel v aplikaci používá jména objektů, nebo atributů, lze zapnout/vypnout na obrazovce nastavení, viz kapitolu 5.9. Při nepoužívání jmen je uživatelský vstup ve formě povinných objektů a atributů chápan pouze jako indexy řádků, respektive sloupců vstupní matice. Vyhledávání pak pracuje přímo s čísly. Pokud je zapnuto používání jmen, vyhledávání pracuje s řetězci a místo přímého porovnávání číselných hodnot ověřuje, zda jméno objektu nebo atributu formálního konceptu obsahuje znaky zadané uživatelem.

Spodní část obrazovky obsahuje seznam nalezených formálních konceptů. Seznam zobrazuje pouze určitou část výsledku vybranou uživatelem v nabídce nad ním. Dále se zde nachází indikátor přibližné polohy formálního konceptu vzhledem k celé matici. Modrý obdélník ohraničuje vnější stranu formálního konceptu. Na pravé straně spodní části obrazovky lze nalezené nebo vybrané formální koncepty označit pro vizualizaci na hlavní obrazovce.

## 5.3 Obrazovka detail souřadnic

Na této obrazovce jsou uživateli zobrazeny informace o jedné nebo více souřadnicích v matici. Lze zde nalézt seznam všech formálních konceptů, které pokrývají danou souřadnici, informace o nich nebo teplotní mapu. Uživateli je tato obrazovka dostupná přes kontextové menu, po kliknutí pravým tlačítkem na pozici s jedničkou na hlavní obrazovce, nebo přes obrazovku detail konceptu, viz kapitolu 5.4. Na obrazovce detail souřadnic, ilustrované obrázkem 5, se postupně nachází:

1. **Seznam souřadnic** – určuje všechny souřadnice, mezi kterými může uživatel přepínat v aktuálním kontextu obrazovky. Souřadnic je více, pokud je obrazovka otevřena za účelem zobrazení detailu společných souřadnic dvou formálních konceptů z obrazovky detail konceptu.
2. **Formální koncepty na dané souřadnici** – udává seznam formálních konceptů vyskytující se právě na uživatelem vybrané souřadnici.
3. **Přibližná lokace konceptu** – ilustruje uživateli, kde přibližně se daný formální koncept nachází vzhledem k celé vstupní matici. Modrý obdélník ohraničuje vnější rozměr formálního konceptu.




Obrázek 5: Obrazovka detail souřadnic

4. **Podoba formálního konceptu** – nachází se na první záložce v hlavní části obrazovky. Pod obrázkem představujícím ilustraci formálního konceptu se nachází výčet všech objektů a atributů. Pro formální koncepty, které se rozprostírají přes více než 10000 řádků, dochází ke škálování, takže ilustrace formálního konceptu může být nepřesná. Škálováním se namapují indexy objektů formálního konceptu do intervalu $[0, 9999]$. Indexy atributů formálního konceptu se neškálují. Takto škálovaný formální koncept je poté vykreslen. Popisky os jsou přizpůsobeny originálním indexům objektů a atributů formálního konceptu.

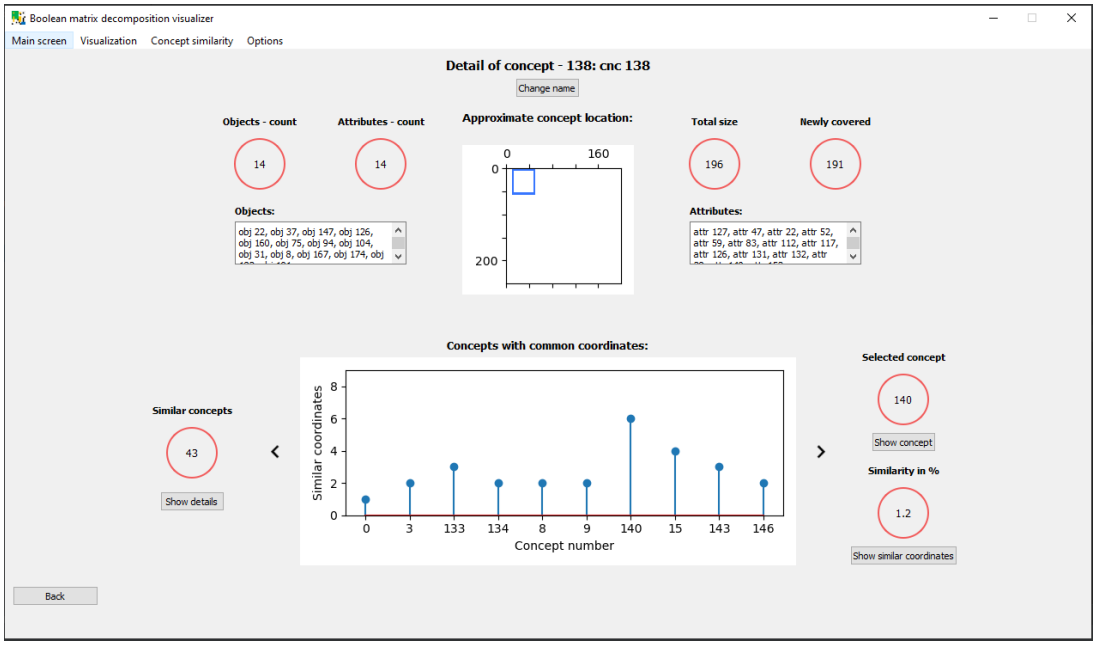
   V nastavení aplikace, viz kapitolu 5.9, pak lze určit, zda obrázek ilustrující formální koncept bude pevně zachovávat čtvercové rozměry pixelu, nebo se přizpůsobí dobré čitelnosti. Tato možnost je uživateli dána proto, že při pevném zachování čtvercových rozměrů, pro formální koncept s mnoha objekty a málo atributy, může být tento obrázek deformovaný.

5. **Teplotní mapa formálních konceptů na dané souřadnici** – je umístěna v druhé záložce. Pro vybranou souřadnici zobrazí teplotní mapu všech formální konceptů jí pokrývající. Souřadnice pokryté více formálními koncepty jsou v teplotní mapě zobrazeny světlejší barvou. Pro teplotní mapu dochází ke stejnému škálování jako pro podobu formálního konceptu.

6. **Detail formálního konceptu** – tvoří poslední záložku a poskytuje podrobné informace o uživatelem vybraném formálním konceptu ze seznamu. Představuje zmenšenou verzi obrazovky detail formálního konceptu, viz kapitolu 5.4.

## 5.4 Obrazovka detail formálního konceptu

Zobrazuje důležité informace o vybraném formálním konceptu, obrazovka je ilustrována obrázkem 6 a postupně se na ní nachází:




Obrázek 6: Obrazovka detail formálního konceptu

1. **Jméno formálního konceptu** – uživatelem definované, nebo importované jméno formálního konceptu. Jméno může být kdykoliv změněno tlačítkem pod ním. V nastavení pak lze zapnout/vypnout používání jmen, viz kapitolu 5.9.

2. **Informace o**:

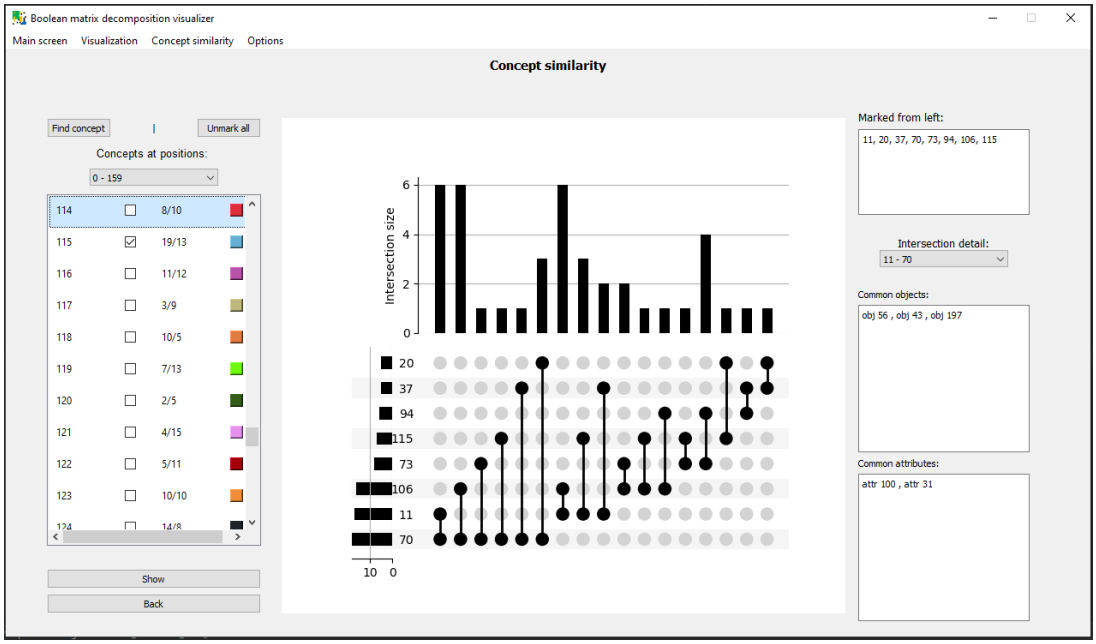
   - Počtu objektů.
   - Počtu atributů.
   - Přibližné poloze vzhledem k celé matici, modrý obdélník ohraničuje vnější rozměr formálního konceptu.
   - Celkové velikosti formálního konceptu ve smyslu obsahovaných jedniček, tedy počet objektů × počet atributů.
   - Počtu nově pokrytých souřadnic. Pokud byl koncept již vizualizován, udává informaci o počtu nově pokrytých jedniček ve vstupní matici právě tímto formálním konceptem, jinak je tento údaj prázdný.

3. **Výčet objektů a atributů**

4. **Údaje o podobných formálních konceptech** – v grafu je zobrazen výčet formálních konceptů, které mají s aktuálním formálním konceptem společné nějaké souřadnice. To znamená, že mají neprázdný průnik objektů a neprázdný průnik atributů. Na ose $x$ grafu je uveden jednoznačný identifikátor formálního konceptu, na ose $y$ počet společných souřadnic. Na levé straně od grafu se zobrazuje počet takových formálních konceptů, kliknutím na tlačítko pod ním lze přejít na obrazovku podobnost konceptů, viz kapitolu 5.5. Grafem se lze pohybovat šipkami umístěnými po levé a pravé straně. Kliknutím na modré kolečko v grafu lze vybrat formální koncept mající společné souřadnice s aktuálním. Při takovém výběru dojde k aktualizaci informace o vybraném formálním konceptu z grafu a údaje o velikosti průniku formálních konceptů vzhledem k jejich velikosti. Po kliknutí na tlačítko pod informací o vybraném formálním konceptu z grafu lze přejít na detail vybraného konceptu. Pokud uživatel klikne na tlačítko pod procentuálním údajem podobnosti, lze přejít na obrazovku detail souřadnic a prohlédnout si všechny společné souřadnice těchto dvou formálních konceptů.

## 5.5 Obrazovka podobnost konceptů

Zobrazuje informace o podobnosti až deseti formálních konceptů na základě výběru uživatele. Tato obrazovka je zobrazena na obrázku 7 a postupně se na ní nachází:




Obrázek 7: Obrazovka podobnost konceptů

1. **Seznam všech formálních konceptů** – setřízený vždy podle jednoznač-

ných identifikátorů formálních konceptů, z tohoto seznamu uživatel vybírá formální koncepty, jejichž podobnost chce znát. Funguje na stejném principu jako seznam na hlavní obrazovce, tedy vždy zobrazí pouze zvolenou část formálních konceptů v nabídce nad ním. Obrazovka podobnosti konceptů může být otevřena třemi způsoby, které mají vliv na tento seznam:

- Z hlavní obrazovky s prázdnou množinou označených formálních konceptů.
- Z hlavní obrazovky s označenými formálními koncepty, které jsou na hlavní obrazovce vybrány pro vizualizaci.
- Z obrazovky detail formálního konceptu, pak budou označeny všechny formální koncepty mající neprázdný průnik souřadnic s formálním konceptem z obrazovky detail formálního konceptu. Takových formálních konceptů může být více než deset, tedy uživatel musí některé ručně odznačit.

Uživatel může v seznamu rychle vyhledat formální koncept na základě jednoznačného identifikátoru pomocí tlačítka nad seznamem. Také může odznačit všechny formální koncepty v seznamu. Po kliknutí pravým tlačítkem na formální koncept lze přejít na obrazovku detail formálního konceptu.

2. **Výčet označených formálních konceptů**

3. **Graf podobnosti** – pro označené formální koncepty zobrazuje údaje o společných souřadnicích. Z obrázku 7 lze například vyčíst, že formální koncepty 11 a 70 mají 6 společných souřadnic.

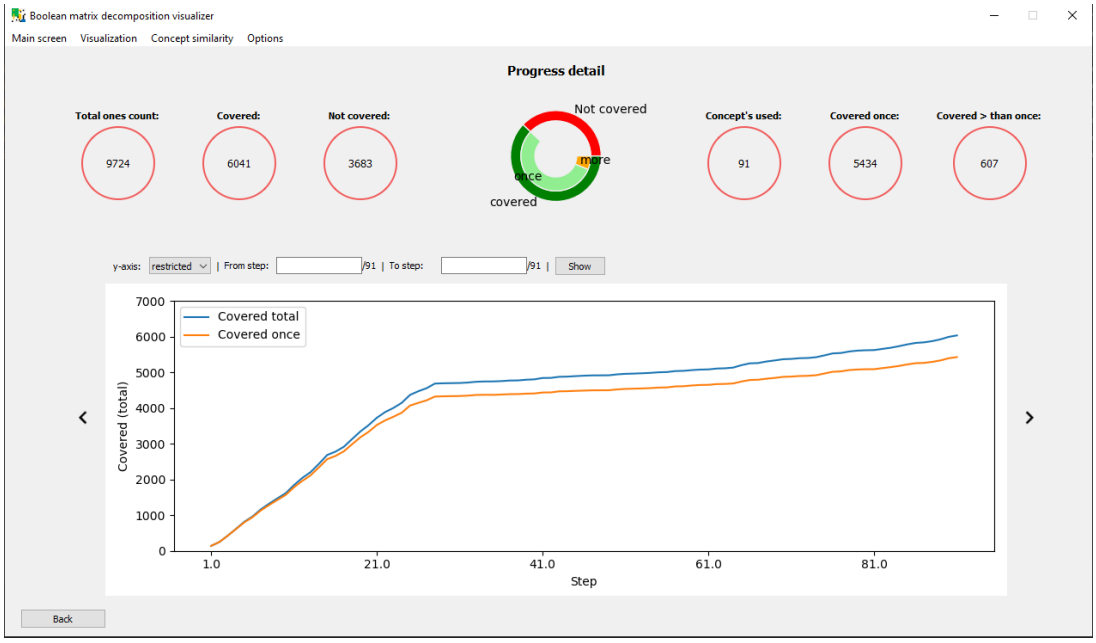
4. **Detail o společných souřadnicích** – uživatel má možnost, vybrat jednu konkrétní dvojici formálních konceptů, která má neprázdný průnik souřadnic ilustrovaný v grafu. Pro tuto dvojici je pak zobrazen výčet společných objektů a společných atributů.

## 5.6 Obrazovka detail průběhu

Na této obrazovce jsou uživateli k dispozici údaje o průběhu vizualizace. Lze zde sledovat jednotlivé kroky vizualizace a informace vázané k těmto krokům, jako je například použitý formální koncept nebo celkový počet pokrytých jedniček vstupní matice v daném kroku. Obrazovka detail průběhu je ilustrována obrázkem 8 a postupně se na ní nachází:

1. **Záhlaví s informacemi o průběhu** – kromě údaje o celkovém počtu jedniček ve vstupní matici dále obsahuje informace vztahující se ke konkrétnímu kroku vizualizace, tedy:

   - počet již pokrytých jedniček,
   - počet ještě nepokrytých jedniček,




Obrázek 8: Obrazovka detail průběhu

- formální koncept vizualizovaný v daném kroku,
- počet jedniček ve vstupní matici pokrytých pouze jednou,
- počet jedniček ve vstupní matici pokrytých vícekrát.

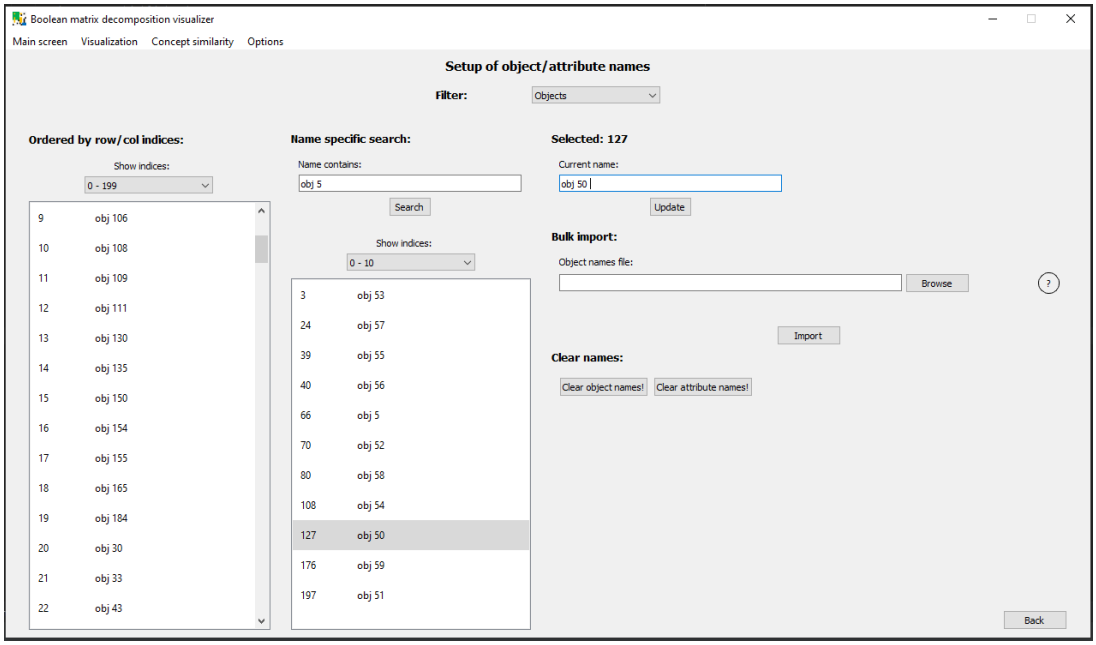
V centru záhlaví se nachází graf vykreslující uvedené informace pro větší přehlednost. Číselné hodnoty v záhlavní je možné přepnout na procentuální v nastavení, viz kapitolu 5.9.

2. **Graf průběhu vizualizace** - zobrazuje historii vizualizace vzhledem k počtu pokrytých jedniček. V záhlaví grafu si může uživatel zvolit, zda maximální hodnota osy $y$ bude dosud dosažený počet pokrytých jedniček, nebo celkový počet jedniček ve vstupní matici. Pokud uživatel klikne na jednu z linií v grafu, aktualizují se informace v záhlavní vzhledem k vybranému kroku. Graf lze také různě přibližovat a oddalovat pomocí zadání rozsahu kroků, které mají být v grafu zobrazeny. Pokud je graf přiblížen, lze se ním pohybovat pomocí vedle umístěných šipek. Za dostatečného přiblížení se zobrazí v grafu jednotlivé body, které lze pomocí pravého tlačítka použít pro otevření detailu formálního konceptu vizualizovaného v daném kroku.

## 5.7 Obrazovka jmen objektů a atributů

Poskytuje přehled o všech definovaných jménech objektů a atributů. Jména lze na této obrazovce jednotlivě měnit nebo importovat více jmen najednou. Uživatel zde také může vyhledávat určité objekty, nebo atributy na základě jejich

jmen. Obrazovka jmen objektů a atributů je ilustrována obrázkem 9 a postupně obsahuje:




Obrázek 9: Obrazovka jmen objektů a atributů

1. **Výběr módu** – dává uživateli na výběr, zda chce spravovat jména pro objekty, nebo pro atributy.

2. **Seznam jmen** – nachází se na levé straně obrazovky, jedna položka v seznamu představuje jeden objekt, nebo atribut. Položka je tvořena odpovídajícím indexem řádku, nebo sloupce v matici, a jménem. Pořadí prvků se různě mění v závislosti na procesu vizualizace.

3. **Vyhledávácí část** – je umístěna ve středu obrazovky a umožňuje uživateli vyhledávat objekty, nebo atributy na základě vstupního řetězce. V seznamu výsledků je opět zobrazena pouze uživatelem vybraná část v nabídce nad ním.

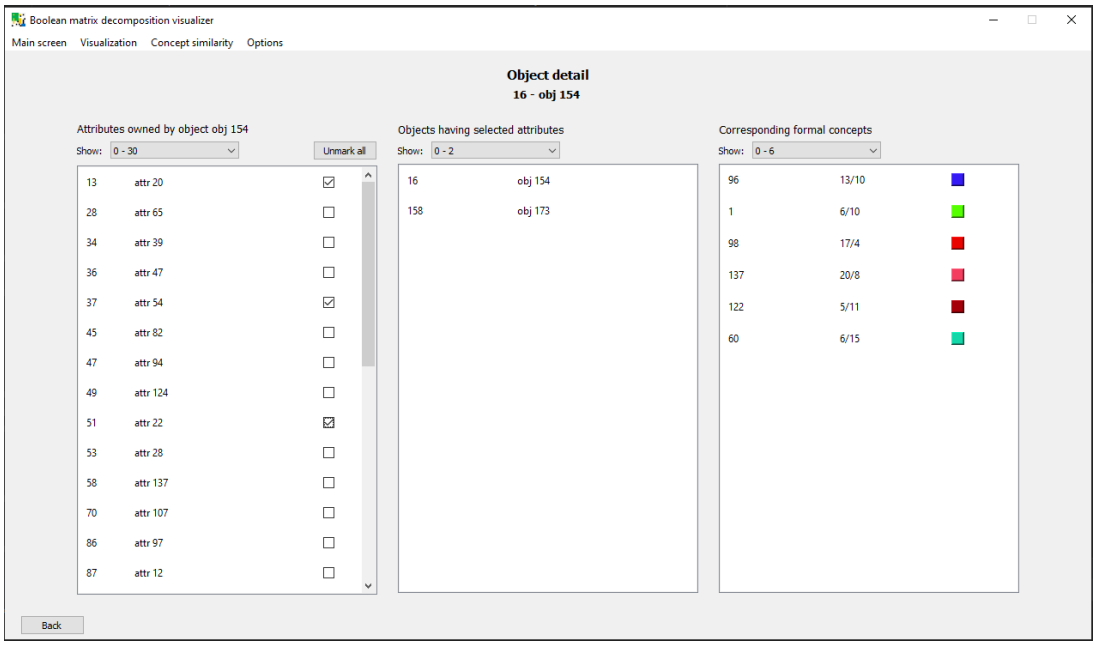
4. **Část pro změnu jména** – je první položka v pravé části obrazovky. Zobrazuje aktuální jméno objektu, respektive atributu vybraného v seznamu všech prvků, nebo v seznamu výsledku vyhledávání. Jméno lze změnit a poté uložit stisknutím tlačítka níže.

5. **Sekce pro import ze souboru** – umožní uživateli nahrát více jmen ze souboru ve specifikovaném formátu. Detail formátu souboru je zobrazen po kliknutí na ikonu otazníku. Pro více informací o importu a exportu dat, viz kapitolu 6.

6. **Tlačítka pro smazání jmen**.

## 5.8 Obrazovka detail objektu

Poskytuje uživateli informace o detailu objektu ve smyslu šipkových operátorů z formální konceptuální analýzy. Podoba obrazovky detail objektu je ilustrována obrázkem 10 a pro objekt $i$ se na ní postupně vyskytuje:




Obrázek 10: Obrazovka detail objektu

1. **Seznam atributů** – atributy, které jsou výsledkem aplikace šipkového operátoru $\uparrow$ na aktuální objekt, tedy $\{i\}^{\uparrow}$. Z tohoto seznamu lze vybrat určitý počet atributů označením zaškrtávacího políčka, označme vybranou množinu atributů $B$.

2. **Seznam objektů** – objekty, které jsou výsledkem $B^{\downarrow}$. Seznam objektů i atributů je opět rozdělen na části, přičemž pouze uživatelem vybraná část je zobrazena.

3. **Seznam formálních konceptů** – v tomto seznamu jsou takové formální koncepty, které mají v extentu minimálně jeden z označených objektů a v intentu minimálně jeden z zobrazených atributů. Pravým kliknutím na formální koncept lze přejít na obrazovku detail formálního konceptu.

Podobně jako obrazovka detail objektu pracuje obrazovka detail atributu. Ta akorát vychází z atributu a používá tomu odpovídající šipkové operátory. Obě obrazovky jsou dostupné přes kontextové menu pravého tlačítka na hlavní obrazovce, kde v matici je hodnota 1.

## 5.9 Obrazovka nastavení

Umožňuje uživateli měnit určité nastavení za běhu aplikace. Postupně je možné měnit:

1. **Velikost okna aplikace** – k dispozici jsou tři předdefinované velikosti (uvedeny v pixelech):
   - $1366 \times 768$,
   - $1600 \times 900$,
   - $1920 \times 1080$.
2. **Aspekt sekundárních grafů** – týká se grafů na obrazovce detail souřadnic. Tento parametr určuje, zda například graf ilustrující podobu formálního konceptu bude zachovávat čtvercové velikosti pixelu (equal), nebo se přizpůsobí lepší čitelnosti (auto). Pro ilustraci viz obrázek 11.
3. **Používání jmen** – lze měnit postupně pro objekty, atributy a formální koncepty. Na výběr má uživatel dvě možnosti, buď používat pouze indexy, nebo jím definovaná jména. Použití jmen objektů a atributů má vliv na způsob vyhledávání na obrazovce výběru formálních konceptů, na popisky os grafů nebo jednotlivé výčty objektů a atributů formálních konceptů. Jména formálních konceptů jsou pak zobrazena na obrazovce detail formálního konceptu a v pomocných textech na hlavní obrazovce.
4. **Používání vláken při krokování** – udává, zda pro krokování vizualizace budou použita samostatná vláka. Lze tak předejít zamrzání okna aplikace, pokud formální koncepty obsahují velké množství objektů, nebo atributů. Naopak pro menší vstupní matice je lepší tuto možnost nevyužívat.
5. **Defaultní počet řádků a sloupců** – určuje iniciální rozměr výřezu matice na hlavní obrazovce. Na tento rozměr bude matice navrácena vždy po stisknutí tlačítka domů na navigačním panelu v pravé části hlavní obrazovky.

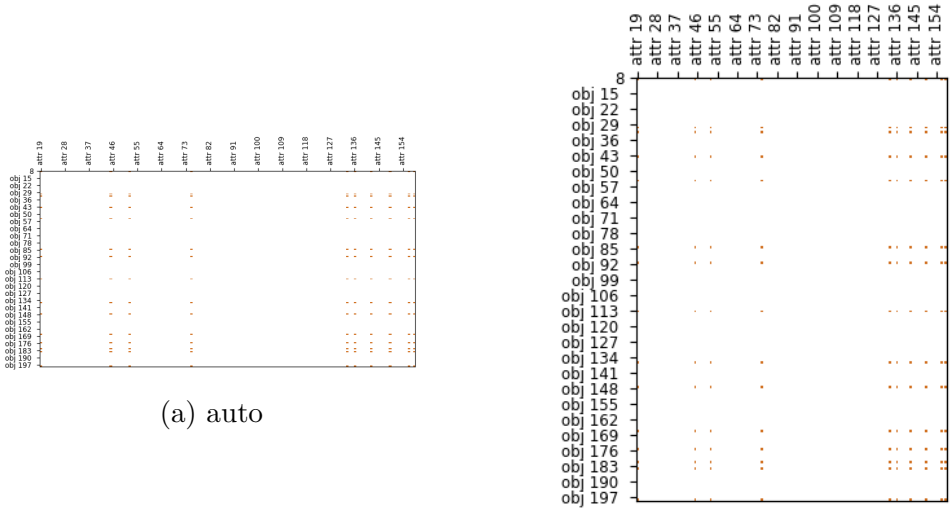
## 5.10 Vybrané implementační aspekty

V této kapitole jsou popsány vybrané implementační detaily aplikace.

### 5.10.1 Vykreslení aktuálního výřezu matice

Obrázek aktuálního výřezu matice na hlavní obrazovce je vykreslen pomocí metody `matshow` knihovny `matplotlib`. Dva hlavní parametry pro tuto metodu jsou:

- **dvourozměrné pole** – reprezentující část vstupní matice, kterou je třeba vykreslit,




(a) auto

(b) equal

Obrázek 11: Dvě možnosti zobrazení sekundárních grafů

- **barevná mapa** – udává vztah mezi hodnotou v matici a její barvou.

Jednotlivé formální koncepty jsou v matici reprezentovány hodnotou o 2 vyšší, než je hodnota jejich jednoznačného identifikátoru, ten je indexován od 0. Číslo 2 je zvoleno proto, že v původní matici jsou pouze binární hodnoty 0 a 1, tedy další možná hodnota, kterou lze do matice zanést, je právě hodnota 2. Výše zmíněná barevná mapa pak přesně udává barvy jednotlivých formálních konceptů. Pro ilustraci vykreslovací metody uvažujme formální koncepty $\langle\{0,1,2\},\{0,1\}\rangle$ mající $id = 0$, $\langle\{1,2\},\{3,4\}\rangle$ s $id = 1$ a barevnou mapu $\langle 0 \to$ □, $1 \to$ ■, $2 \to$ ■, $3 \to$ ■$\rangle$. Na obrázku 13 je ilustrován výsledek aplikace metody `mathsow` s takto definovanými argumenty na matici popsanou obrázkem 12.

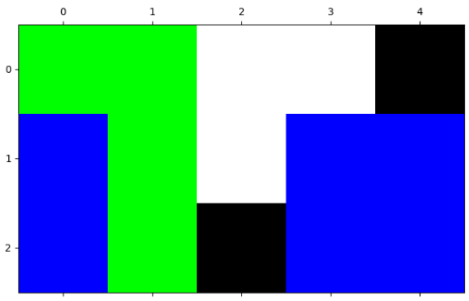
### 5.10.2 Pohyb po aktuálním výřezu matice

Pro plynulejší pohyb po aktuálním výřezu matice je třeba předzpracovávat budoucí vizualizované stavy dopředu. Tedy zajistit dostupnost obrázku co nejdříve okamžiku, kdy uživatel například otočí kolečkem myši. Samotné volání metody `matshow` může trvat i déle než vteřinu.

Z aktuálního výřezu matice je třeba předzpracovávat budoucí stavy pro všechny čtyři možné směry pohybu. Přímo se nabízí použití vláken, kde každé vlákno bude volat vykreslující metodu a předzpracovávat jeden konkrétní směr pohybu. Problém při použití vláken představuje *Global Interpreter Lock(GIL)* jazyka Python. Ten zkráceně řečeno zabraňuje paralelnímu běhu dvou vláken stejného procesu [13]. Tento problém lze vyřešit použitím procesů místo vláken a naplno tak využít procesor s více fyzickými jádry. Celkem jsou použity 4 samostatné procesy, které spolu vzájemně komunikují. Předzpracované obrázky jsou postupně ukládány do sdílené fronty, ze které je pak přečte hlavní proces.

$$\begin{pmatrix} 2 & 2 & 0 & 0 & 1 \\ 3 & 2 & 0 & 3 & 3 \\ 3 & 2 & 1 & 3 & 3 \end{pmatrix}$$

Obrázek 12: Matice pro vykreslení




Obrázek 13: Výsledek vykreslení

Pro správné předzpracování obrázků je důležitá zejména komunikace mezi procesy. Komunikace je založena na sdíleném stavu, ten je tvořen náhodným číslem. Každá akce provedená uživatelem, která jakýmkoliv způsobem změní aktuální výřez matice, změní také sdílený stav. Dílčí procesy zajišťující předzpracování před každou další iterací ověří, zda se sdílený stav změnil. Pokud ano, aktualizují si informace o aktuálním výřezu matice, vyprázdní sdílenou frontu a předzpracování začíná znovu.

### 5.10.3 Práce s velkým objemem dat

Pro účely aplikace byla uvažována maximální velikost vstupní matice v řádu desítek tisíc řádků a stovek sloupců. Booleovská matice mající například 80000 řádků a 1000 sloupců bude v paměti RAM zabírat 8MB při potřebě jednoho bitu na reprezentaci jednoho elementu v matici. Pro vykreslovací účely jsou ale hodnoty v matici přímo závislé na počtu formálních konceptů. Konkrétně, maximální hodnota prvku v matici může být o 2 větší, než je maximální počet formálních konceptů.

Pro reprezentaci elementů vykreslovací matice byl zvolen 32-bitový neznaménkový `integer`. Počet formálních konceptů je tedy shora omezen hodnotou $2^{32} - 2$. Výše uvedená booleovská matice bude nyní v paměti zabírat 256MB. Za běhu aplikace je dále nutné mít přístup k následujícím údajům:

- seznam všech formálních konceptů,
- jména definovaná pro objekty a atributy,
- barvy odpovídající formálním konceptům,
- historie vizualizace – kolik bylo v $i$-tém kroku nově pokrytých jedniček apod.,
- pomocné struktury – zjednodušují určité operace během vizualizace. Mezi tyto struktury patří například slovník udržující informace o formálních konceptech obsahující daný objekt, nebo seznam obsahující podobu formálních konceptů těsně před vizualizací.

Skoro všechny údaje jsou přímo závislé na počtu formálních konceptů. Mohlo by se stát, že dané zařízení nebude mít dostatek paměti RAM na uložení všech potřebných informací. Pro zabránění tomuto problému jsou potřebné informace ukládány přímo na disk a namapované do paměti pomocí metody `memmap` knihovny `numpy`. Díky tomuto přístupu je možné mít na disku uložený velký soubor, se kterým lze pracovat jako s polem. Do paměti RAM lze poté načítat pouze tu část, která je právě požadovaná aplikací. Další výhodou tohoto přístupu je perzistence dat, viz kapitolu 6.

### 5.10.4 Problém zpětné vizualizace

Jak vizualizace postupuje a více formálních konceptů je vyobrazeno v aktuálním výřezu matice, může se stát, že jeden formální koncept při vykreslení překryje část jiného. Tato situace je ilustrována například na obrázku 13. Pokud uživatel postupuje pouze dopředu, tedy vizualizuje další formální koncepty, je vše v pořádku. Problém nastává, pokud chce uživatel provést krok zpět. Při odbarvení posledního vykresleného formálního konceptu je třeba zjistit, zda na některých pozicích překrýval jiný formální koncept a pokud ano, použít pro danou pozici odpovídající barvu.

Tedy pro každou souřadnici vstupní matice je třeba udržovat historické údaje o tom, jaké formální koncepty na ni jsou postupně vykresleny. Z důvodu velikosti historických dat je opět zvolen přístup, který tato data neudržuje v paměti RAM, ale přímo na pevném disku. Tentokrát nelze použít metodu `memmap` knihovny `numpy`, protože objekt na disku, se kterým metoda pracuje, je typu pole. Pro správnou alokaci souboru je třeba znát maximální počet formálních konceptů pokrývajících libovolnou souřadnici matice. Zjištění této hodnoty pro větší matice je značně neefektivní operace.

Pro udržování historických dat byla zvolena `sqlite3` databáze s tabulkou obsahující dva atributy:

- **Souřadnice** – řetězec ve tvaru "$x \times y$", kde $x$ udává index řádku a $y$ index sloupce. Tento atribut je primárním klíčem tabulky a je na něm vytvořen index pro rychlejší vyhledávání.
- **Historie** – řetězec ve tvaru "$cnc_1 - cnc_2 - ... - cnc_n$", kde $cnc_i$ pro $i = 1, ..., n$ představuje jednoznačný identifikátor formálního konceptu a $cnc_n$ indikuje poslední vykreslený formální koncept na určité souřadnici.

Po odbarvení vizualizovaného formálního konceptu je vždy aktualizována historická tabulka. Zjistit formální koncept pro vykreslení na dané souřadnici lze poté následovně:

1. Dotázat se databáze na řetězec historie pro určitou souřadnici.
2. Z obdrženého řetězce rozparsovat pouze část za poslední pomlčkou.

Pro veškerou práci s databází je využíváno hromadné zpracování pomocí metody `executemany` modulu `sqlite3`, za účelem minimalizace počtu interakcí s databází a dosažení tak větší rychlosti.

# 6 Import a export dat

Aplikace popsaná v sekci 5 pracuje s perzistentními daty. To jsou taková data, která přežívají výpočetní proces. Uživatel tedy může do aplikace naimportovat určitá data a ta jsou pak použity při každém dalším startu aplikace, dokud nejsou importována data jiná. Při každém dalším startu aplikace se nezohledňují změny, které uživatel provedl v předešlém spuštění, s výjimkou změny barev. Tedy pokud uživatel při jednom spuštění aplikace změní pořadí vizualizace formálních konceptů a barvy příslušné některým z nich. Pak při dalším spuštění aplikace bude pořadí formálních konceptů resetováno do importované podoby, nicméně změna barev zůstane zachována. Dále jsou popsány informace, které lze do aplikace importovat, a struktury odpovídajících souborů. Všechny soubory pro import a export jsou ve formátu `txt`.

**Formální koncepty**
Soubor představující seznam formálních konceptů. Jedná se o jediný povinný soubor při procesu importu dat do aplikace. Z tohoto seznamu je automaticky zkonstruována vstupní matice, na které probíhá vizualizace. Soubor pro import formálních konceptů musí mít následující strukturu:

- Na prvním řádku je informace o rozměrech matice ve tvaru "$w$x$h$", kde $w$ představuje počet sloupců a $h$ počet řádků.
- Maximální velikost matice je omezena na 100000 řádků a 1000 sloupců.
- Informace o jednom formálním konceptu jsou uvedeny na jednom řádku, za formálním konceptem následuje znak nového řádku.
- Extent a intent formálního konceptu jsou na řádku odděleny znakem "|" bez mezer.
- Jednotlivé objekty, respektive atributy, jsou odděleny znakem "," bez mezer.
- Výčet objektů na řádku předchází výčtu atributů.

**Soubory se jmény**
Do této kategorie spadají tři soubory definující postupně jména objektů, atributů a formálních konceptů. Všechny tři soubory se jmény jsou při importu nepovinné. Pokud je uživatel neuvede, příslušná jména nebudou importována. Nicméně je lze definovat později v aplikaci, viz kapitolu 5.7. Textové soubory pro import jmen musí mít následující strukturu:

- Jednotlivé informace o jménech jsou odděleny znakem "," bez mezer.

- Každá informace o jménu má formát "*index:jméno*".
- *Index* v předchozím bodu značí:
  - číslo řádku v matici odpovídající pojmenovávanému objektu – pro soubor se jmény objektů,
  - číslo sloupce v matici odpovídající pojmenovávanému atributu – pro soubor se jmény atributů,
  - jednoznačný identifikátor formálního konceptu – pro soubor se jmény formálních konceptů.
- *Jméno* značí řetězec určující pojmenování daného objektu, atributu, nebo formálního konceptu.
- Řetězec *jméno* je omezen délkou 30 znaků.
- Na jednom řádku v souboru může být proměnné množství informací o jménech, doporučené množství je 1-100.
- Je možné uvést jména pouze pro vybrané objekty, atributy nebo formální koncepty.

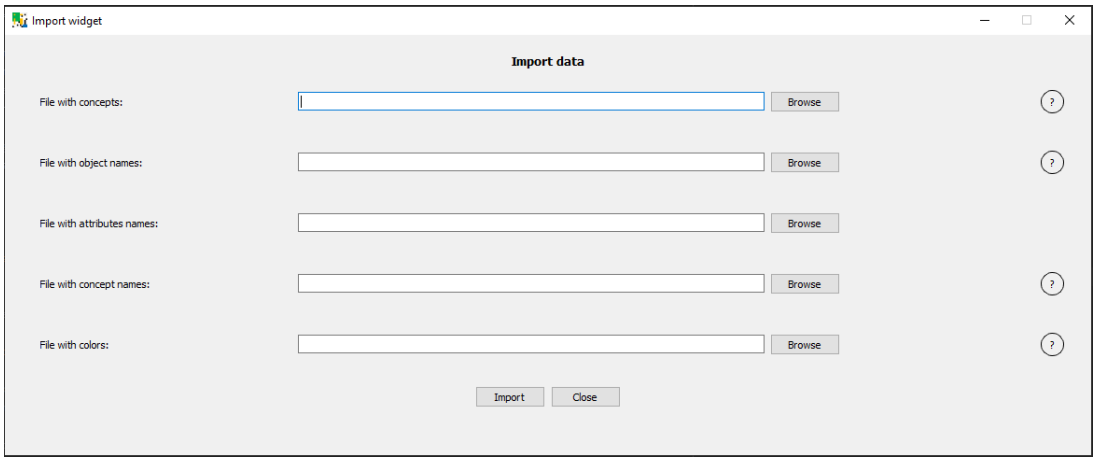
**Soubor s barvami**
Definuje barvy jednotlivých formálních konceptů. Soubor s barvami je při importu dat nepovinný. Pokud jej uživatel nevyplní, jsou barvy vygenerovány náhodně a mohou být změněny později v aplikaci, viz kapitolu 5.1. Soubor definující barvy formálních konceptů musí mít následující strukturu:

- Jednotlivé informace o barvách formálních konceptů jsou odděleny znakem "|" bez mezer.
- Jedna informace o barvě formálního konceptu má formát "*index:r,g,b*", kde *index* představuje jednoznačný identifikátor formálního konceptu, dále *r,g,b* indikují hodnoty barevného schématu RGB. Tedy $r, g, b \in \{0, 1, ..., 255\}$.
- Jednotlivé informace o barvách nemohou být rozděleny znakem nového řádku.
- Na jednom řádku může být proměnné množství informací o barvách formálních konceptů, doporučení množství je 1-100.
- Je možné uvést pouze barvy pro některé formální koncepty, zbytek bude doplněn náhodně.

Příklady jednotlivých vstupních souborů lze nalézt na obrazovce import dat, viz kapitolu 6.1. Konkrétní vstupní soubory použité pro inicializaci aplikace lze nalézt v adresářové struktuře přiloženého CD, viz přílohu B.

## 6.1 Obrazovka pro import dat

Je samostatná obrazovka zajišťující pouze import dat v předepsaném formátu do aplikace. Při importu dat jsou konstruovány perzistentní datové struktury, které usnadňují některé výpočetní operace při procesu vizualizace. Import dat tedy může trvat delší dobu, zvláště pro velké množství formálních konceptů. Obrazovka pro import dat je ilustrována obrázkem 14.




Obrázek 14: Obrazovka pro import dat

Na obrazovce pro import dat se nachází vstupní pole pro jednotlivé importované soubory popsané výše. Po kliknutí na tlačítko otazníku na pravé straně obrazovky lze zjistit přesné požadavky na formát konkrétního importovaného souboru spolu s jednoduchým příkladem.

Obrazovka pro import dat musí být spuštěna samostatně. Před spuštěním je nutné ukončit hlavní aplikaci. Před opětovným spuštěním hlavní aplikace je nutné zavřít tuto obrazovku. Podrobnosti o spuštění jsou uvedeny v příloze A.

## 6.2 Export dat

Aktuální konfiguraci aplikace lze exportovat pomocí hlavního menu. Exportovanou konfiguraci postupně tvoří:

- Soubor s formálními koncepty – zde je brán v potaz aktuální stav vizualizace a pořadí formálních konceptů pro vizualizaci. Tedy formální koncepty mohou být exportované ve stavu, po slití algoritmem 1 a v pozměněném pořadí.
- Soubory se jmény objektů, atributů a formálních konceptů.
- Soubor s barvami formálních konceptů.

# 7 Limity a omezení aplikace

V této kapitole jsou popsána jednotlivá omezení a neduhy aplikace spolu s jejich vysvětlením.

**Import dat**
Import dat je prováděn samostatně z toho důvodu, že hlavní aplikace je pak inicializována z importovaných dat. Přepsání všech importovaných dat v již běžící aplikaci by bylo pomalejší než její opětovné spuštění. Časová náročnost importu dat je nejvíce ovlivněna skutečností, že se z importovaných formálních konceptů konstruuje vstupní matice. Je tedy nutné projít všechny formální koncepty a zjistit které, souřadnice v matici pokrývají. Proces konstrukce matice je paralelizován, nicméně může trvat delší dobu.

Konstrukce vstupní matice z 1000 formálních konceptů majících průměrně 300 objektů a 75 atributů trvá okolo 5 minut. Výsledná matice při testování měla 30000 řádků a 1000 sloupců.

**Start aplikace**
Při startu aplikace jsou konstruovány struktury, usnadňující výpočty při procesu vizualizace. Mezi tyto struktury patří například slovníky, usnadňující přístup k formálním konceptům, obsahující v extentu daný objekt, nebo v intentu atribut. Dále jsou resetována data z předchozího běhu aplikace a inicializovány procesy pro předzpracování. Díky těmto akcím může spuštění aplikace pro velká data trvat až 20 vteřin.

**Velikost aktuálního výřezu matice**
Velikost aktuálního výřezu matice na hlavní obrazovce je omezena na 10000 řádků, sloupce omezeny nejsou. Omezení je zavedeno jako prevence před chybami souvisejícími s nedostatkem paměti RAM. K těm dojde v případě, když se knihovna `matplotlib` snaží alokovat paměť pro vykreslení aktuálního výřezu matice, který je moc velký. Problém je řešitelný zvětšením maximální velikosti stránkovacího souboru operačního systému. Nicméně to je netriviální zásah do systému, který po uživateli nelze požadovat.

**Obrazovka detail souřadnic**
Vykreslení obrazovky detailu souřadnic může pro větší matice trvat déle, protože je třeba inicializovat a škálovat graf podoby formálního konceptu a teplotní mapu. Vizualizace těchto grafů se zpomaluje s počtem a velikostí formálních konceptů, pokrývajících danou souřadnici.

# 8 Použité technologie

**Python3**
Je multiplatformní vysokoúrovňový skriptovací jazyk. Podporuje různá programovací paradigmata, jako je například funkcionální nebo objektové. Dále podporuje dynamické typování, tedy není nutné deklarovat datové typy pro jednotlivé proměnné. Zároveň jedna proměnná může za běhu programu nabývat hodnot více datových typů. Python je jazykem interpretovaným, tedy pro spuštění aplikace je potřeba interpret jazyka. Při vývoji aplikace byla použita verze Pythonu 3.7.3 a využito bylo následujících knihoven:

- numpy – pro práci s maticemi a datovými soubory,
- pyqt5 – k tvorbě uživatelského rozhraní,
- matplotlib – slouží pro kreslení grafů a obecné vizualizaci informací,
- pandas – knihovna pro práci s daty,
- upsetplot – vyobrazení průniku více množin,
- psutil – multiplatformní knihovna sloužící pro přístup k informacím o běžících procesech.

Některé z uvedených knihoven obsahují reference na další knihovny. Seznam tedy neobsahuje přesně všechny knihovny, které jsou nutné pro spuštění aplikace. Nicméně po instalaci výše uvedených budou ostatní potřebné knihovny doinstalovány automaticky. Instalace knihoven je popsána v příloze A.

**PyCharm**
Multiplatformní vývojové prostředí zejména pro jazyk Python vyvinuté společností JetBrains. Umožňuje uživateli například debugovat[4] existující kód, také automaticky doplňuje kódová slova nebo podporuje verzovací systémy. Výhodou vývojového prostředí Pycharm je možnost mít pro každý projekt vlastní virtuální interpret jazyka Python, obsahující pouze potřebné knihovny pro daný projekt. Kromě jazyka Python lze využít Pycharm také například pro webové frameworky Django a Angular.

[4]krokování programu, například za účelem hledání chyb

# Závěr

V rámci diplomové práce byl čtenář seznámen s problematikou dekompozice booleovských matic. Dále byl popsán princip vizualizace výsledku booleovské dekompozice jako formálních konceptů na vstupních datech. Princip vizualizace byl spolu s dalšími nástroji sloužícími pro analýzu booleovské dekompozice matic implementován do multiplatformní aplikace.

# Conclusions

In the diploma thesis the reader was acquainted with problematic of boolean matrix decomposition. The method of how the result of boolean matrix decomposition as formal concepts are visualized on input data was described. This method along with other features for analysis of boolean matrix decomposition was implemented in multiplatform application.

# A Zprovoznění aplikace

## A.1 Instalace Pythonu a knihoven

Před spuštěním aplikace je nedříve nutné nainstalovat Python, pro účely aplikace doporučuji nainstalovat verzi 3.7.3 a novější, stahovat lze z oficiální stránky. Aplikace by měla být spustitelná na jakékoliv verzi Pythonu 3, nicméně tato funkcionalita nebyla testována. Po instalaci Pythonu je nutné nainstalovat knihovny uvedené v kapitole 8. Instalace knihoven se provádí v příkazové řádce pomocí příkazu: `pip install "jmeno"`. Například pro knihovnu pyqt5 je příkaz pro její instalaci následující:

```
pip install pyqt5
```

Pokud se na systému Windows 10 při spuštění příkazu pro instalaci vyskytne problém, že příkaz `pip` je neznámý, je nutné přidat cestu k interpretu Pythonu do systémové proměnné `Path`.

## A.2 Hlavní aplikace

Pro spuštění hlavní aplikace i obrazovky pro import dat je nutné, zkopírovat adresářovou strukturu aplikace na pevný disk počítače.

**Systémové požadavky**
Níže jsou popsány doporučené systémové požadavky pro běh aplikace:

- pamět RAM 4 GB a více,
- pevný disk typu SSD, nebo M2 s alespoň 5 GB volného místa,
- minimální rozlišení monitoru 1366 × 768 pixelů.

**Spuštění aplikace**
V adresářové struktuře aplikace je třeba otevřít složku `application`, spustit příkazovou řádku z této složky a provést následující příkaz:

- `python Application.py` – pro Windows,
- `python3 Application.py` – na Linuxových systémech, testováno na Debianu verze 10.

## A.3 Spuštění obrazovky pro import dat

Pro spuštění obrazovky importu dat je třeba otevřít v adresářové struktuře aplikace složku `application`. Poté z této složky otevřít příkazovou řádku a v závislosti na operačním systému provést příkaz:

- `python ImportScreen.py` – pro Windows,
- `python3 ImportScreen.py` – na Linuxových systémech, testováno na Debianu verze 10.

# B Obsah přiloženého CD/DVD

**`src/`**
: Adresářová struktura aplikace pro vizualizaci booleovské dekompozice matic. Obsahuje všechny zdrojové kódy.

**`doc/`**
: Text práce ve formátu PDF včetně zdrojových souborů potřebných pro vygenerování PDF dokumentu.

**`data/`**
: Příklad vstupních dat pro aplikaci, při prvním spuštěním je aplikace inicializovaná s těmito daty.

**`libs.txt`**
: Textový soubor obsahující reference na dokumentaci k používaným knihovnám.

# Literatura


[1] GANTER, B.; WILLIE, R. Formal concept analysis: Mathematical Foundations. Springer, 1999

[2] BELOHLAVEK, R. Introduction to Formal Concept Analysis, výukový text. Olomouc, 2008. [online].
Dostupné z: https://phoenix.inf.upol.cz/esf/ucebni/formal.pdf

[3] BELOHLAVEK, R.; VYCHODIL, V. Discovery of optimal factors in binary data via a novel method of matrix decomposition. In: Journal of Computer and System Sciences, 76(1)(2010), 3-20.

[4] MIETTINEN, P.; MIELIKÄINEN, T.; GIONIS, A.; DAS, G.; MANNILA, H. The discrete basis problem. In: IEEE Transactions on Knowledge and Data Engineering 20(10)(2008), 1348-1362. [online].
Dostupné z https://people.mpi-inf.mpg.de/ pmiettin/papers/dbp.pdf

[5] JANOSTIK R. Metody a algoritmy rozkladů Booleovských matic, diplomová práce. Olomouc, 2015. [online].
Dostupné z: https://theses.cz/id/3p9j9c/.

[6] SIPSER, M. Introduction to the Theory of Computation. PWS Publishing Company, Boston, MA, 1997. ISBN 0-534-94728-X

[7] STOCKMEYER, L. J. The set basis problem is NP-complete. IBM ResearchReport RC5431. 1975

[8] GOLUB, G. H,; VAN LOAN, C. F. Matrix Computations, 3rd ed. Baltimore, MD: Johns Hopkins, 1996.

[9] LEE, D.; SEUNG, H. Learning the parts of objects by Non-negative Matrix Factorization. Nature401(1999), 788–791.

[10] LEEUW, D. Principal component analysis of binary data. Application to roll-call analysis [online]. Dostupné z: https://escholarship.org/uc/item/6hs9x308

[11] TANG, F.; TAO, H. Binary Principal Component Analysis. In: BMVC. 2006, 377-386.

[12] KUZNETSOV, S.; OBIEDKOV, S. Comparing performance of algorithms for generating concept lattices. J. Exp. Theor. Artif. Intell. 14(2)(2002), 189-216.

[13] SUMMERFIELD, M. Python in Practice: Create Better Programs Using Concurrency, Libraries, and Patterns. Addison-Wesley, 2013