PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

DIPLOMOVÁ PRÁCE

Faktorové kódování obrazu

2013 David Beer

Anotace

Faktorová analýza hledá v datech obecnější souvislosti, tzv. faktory, a na základě nich tvoří rozklad původních dat. Jedním možným přístupem je použití formálních konceptů jako faktorů. Tato metoda vede na dobrou interpretaci rozkladu, zachycující vztah mezi objekty a faktory a vztah mezi faktory a původními atributy. Snahou je minimalizovat velikost rozkladu (počet faktorů). Lepších výsledků lze dosáhnout v případě, kdy není vyžadována úplná přesnost rozkladu. Zabývám se vyžitím této metody v oblasti komprese obrazu. Popisuji dva možné algoritmy pro hledání vhodných konceptů tak, aby byl rozklad co nejpřesnější a zároveň co nejmenší. Dále zkoumám vlastnosti metody, jako je vliv volby struktury reziduovaného svazu, efektivita metody pro různé typy obrazových dat a jak rychle se zvyšuje přesnost rozkladu přidáváním dalších faktorů.

Děkuji Doc. RNDr. Vilému Vychodilovi, Ph.D. za jeho čas a trpělivost při zodpovídání nespočet dotazů a zřízení a údržbu vzdáleného serveru, na kterém jsem provedl většinu výpočtů.

Obsah

1. Faktorová analýza **6**
1.1. Formální koncepty jako faktory 6
1.2. Aplikace při kompresi obrazu 9
1.3. Dosavadní experimenty 9

2. Algoritmy **10**
2.1. Set Cover 10
2.2. Promising Columns 11
2.2.1. Implementační optimalizace 13

3. Reprezentace reziduovaných svazů **17**

4. Použité nástroje **19**

5. Výsledky **20**
5.1. Časová a paměťová efektivita algoritmů 20
5.1.1. Set cover – vliv struktury 20
5.1.2. Promising columns – vliv struktury 22
5.1.3. Vzájemné srovnání algoritmů 25
5.1.4. Paměťová náročnost 26
5.1.5. Shrnutí efektivity algoritmů 26
5.2. Kvalita rozkladů 27
5.2.1. Úplné rozklady 27
5.2.2. Částečné rozklady 27
5.2.3. Průběh rozkladu 29
5.3. Vhodné a nevhodné typy obrázků z hlediska komprese 35
5.4. Snížení počtu stupňů 39
5.5. Podvzorkování obrázků 44

6. Diskuze **47**

Závěr **49**

Conclusions **50**

Reference **51**

A. Dokumentace programu gfd **52**
A.1. Použití 52
A.2. Popis 52
A.3. Parametry 52
A.3.1. Parametry ovlivňující kompresní mód 52

A.3.2. Parametry ovlivňující dekompresní mód 53
A.4. Příklady použití 54

B. Obsah přiloženého CD **55**

Seznam obrázků

1. Rozmístění idempotentů struktur densemid a sparsemid 24
2. Průběh rozkladu z pohledu kvality 31
3. Průběh rozkladu z pohledu počtu faktorů 32
4. Rekonstrukce klasického obrázku pro různý počet faktorů 33
5. Rekonstrukce klasického obrázku pro různý počet faktorů 34
6. Rekonstrukce uměle vytvořeného obrázku pro různý počet faktorů 34
7. Postupné překrývání faktorů 35
8. Příklady vhodných obrázků 36
9. Faktory struktur Łukasiewicz a minimum pro obrázek art83 . . . 37
10. Faktory obrázků art41 a art82 38
11. Příklady nevhodných obrázků 39
12. Rekonstrukce obrázků z úplných rozkladů pro různé redukční faktory 42
13. Rekonst. ob. z část. rozkladů s 50 faktory pro různé redukční faktory 43
14. Schéma procesu podvzorkování 44
15. Srovnání blízkosti obrázků s různým stupněm podvzorkování . . . 45
16. Dvě různé metody zpětného zvětšení 46
17. Rekonstrukce obrázku z rozkladu velikosti 5 kB 46

Seznam tabulek

1. Set cover – vliv struktury 21
2. Set cover – vliv struktury na počet faktorů 22
3. Promising columns – příklad 23
4. Promising columns – vliv struktury na dobu výpočtu 25
5. Vzájemné srovnání algoritmů set cover a promising columns . . . 26
6. Úplné rozklady – vliv struktury 27
7. Částečné rozklady (pokryté pozice) – vliv struktury 28
8. Částečné rozklady (blízkost) – vliv struktury 29
9. Průběh rozkladu z pohledu kvality (pokryté pozice) 30
10. Průběh rozkladu z pohledu kvality (blízkost) 30
11. Průběh rozkladu z pohledu počtu faktorů (pokryté pozice) 31
12. Průběh rozkladu z pohledu počtu faktorů (blízkost) 32
13. Počty faktorů úplných rozkladů vhodných a nevhodných obrázků. 35
14. Vliv redukce stupňů na rozklady 41
15. Vliv podvzorkování obrázku na rozklady 45

1. Faktorová analýza

Jedním z důležitých problému současné doby, kdy nám zautomatizované procesy generují velké množství dat, která jsou pro člověka příliš objemná, je redukce dimenzionality dat. Ve vstupních datech, ve kterých jsou sledované objekty (např. organismy) popsané pomocí pro daný účel významných atributů (např. míra výskytu v jednotlivých ekosystémech), se snažíme nalézt určité obecnější souvislosti (např. organismy s vysokou mírou výskytu v konkrétních deseti ekosystémech jsou organismy dobře adaptované na chladné podnebí). Redukce dimenzionality abstrahuje od specifik problému (mezi konkrétními deseti ekosystémy nejsou pro daný účel významné rozdíly, jejich společným faktorem ovšem je, že se všechny nacházejí v chladném podnebí). To zjednodušuje data pro opakované zkoumání a umožňuje v datech nalézt důležité souvislosti, které nebyly v původních datech patrné.

Faktorová analýza je jedním z přístupů k redukci dimenzionality dat. Vstupní data si lze představit jako matici popisující vztah mezi objekty a atributy. Řádky odpovídají objektům a sloupce atributům. Hodnota v i-tém řádku a v j-tém sloupci popisuje v jaké míře nabývá i-tý objekt j-tého atributu. Faktorová analýza se snaží v datech nalézt faktory a pomocí nich rozložit vstupní matici na dvě matice, první zachycující vztah mezi objekty a faktory a druhá zachycující vztah mezi faktory a atributy. Z první matice například vyčteme, že daný organismus se téměř nevyskytuje v suchých podnebích a z druhé matice vyčteme do jaké míry má daný ekosystém suché podnebí. Nalezeným faktorem v tomto případě byl „ekosystémy se suchým podnebím“. Původní data lze získat vhodně definovaným maticovým součinem těchto matic. Součin zde hraje roli kombinačního mechanismu mezi objekty a atributy přes faktory. Problém faktorové analýzy je tedy dán rovnicí

$$I_{m\times n} = A_{m\times k} \circ B_{k\times n},$$

kde I je vstupní matice o m objektech a n atributech, A je matice popisující vztah mezi objekty a faktory a B je matice popisující vztah mezi faktory a atributy. Číslo k je počet nalezených faktorů. Cílem je, aby počet faktorů byl co nejmenší. Zejména by mělo být faktorů mnohem méně než původních atributů, jinak by nedošlo k požadované redukci dimenzionality a nalezení obecnějších souvislostí v datech. Pokud nejsou vyžadovány úplně přesné rozklady, mohou být přijatelné i situace, kdy má úplný rozklad více faktorů než je původních atributů, pokud lze najít částečný rozklad s dostatečně nízkým počtem faktorů, tak aby rozklad reprezentoval matici, která je dostatečně blízká původní matici.

1.1. Formální koncepty jako faktory

Jedním z možných přístupů k faktorové analýze (FA) je využití nástrojů formální konceptuální analýzy (FCA), viz. [7]. Pojem konceptu z FCA je totiž blízký

pojmu faktoru z FA. V tomto případě je potřeba, aby data v maticích pocházela z konečné uspořádané množiny. Jde tedy o hodnoty komparativního charakteru neboli stupně. Tento požadavek je přirozený, protože data jsou často výstupem různých dotazníků, testů či měření, pro které lze zavést stupnici hodnot.

Jelikož budeme pracovat se stupni, budeme potřebovat tzv. strukturu reziduovaného svazu. Reziduované svazy jsou struktury vyskytující se ve fuzzy logice. Reziduovaný svaz $\langle L, \wedge, \vee, \otimes, \rightarrow, 0, 1\rangle$ rozšiřuje klasický ohraničený svaz $\langle L, \wedge, \vee, 0, 1\rangle$ o binární operace $\otimes$ a $\rightarrow$ na množině stupňů L, které zobecňují logické funkce konjunkce a implikace z klasické logiky na množinách stupňů jiných než $\{0, 1\}$ a vzájemně splňují tzv. podmínku adjunkce, která zajišťuje platnost odvozovacího pravidla *modus ponens*. Pro přesnou definici reziduovaného svazu odkazuji na definici 1.3 v [3]. Volba konkrétní struktury reziduovaného svazu závisí na druhu zkoumaných dat.

Při počítání skalárního součinu řádku a sloupce v maticovém násobení z rovnice

$$I = A \circ B$$

se místo násobení použije operace $\otimes$ a místo součtu se použije operace $\vee$. Prvky matic jsou z množiny L. Prvek $I_{i,j}$ se tedy spočítá jako

$$I_{i,j} = \bigvee_{1 \leq l \leq k} (A_{i,l} \otimes B_{l,j}).$$

Operace $\otimes$ musí být vzhledem k operaci $\vee$ distributivní, což zajišťuje, že daná struktura bude reziduovaný svaz (viz. Theorem 1.22 z [3]).

Rozklad matice I vytvoříme z formálních konceptů. Formální koncept je dvojice fuzzy množin. První složka se označuje jako extent a zachycuje stupeň, v jakém se koncept vztahuje na jednotlivé objekty (v jakém stupni se daný ekosystém nachází v chladném podnebí). Druhá složka se označuje jako intent a zachycuje stupeň, v jakém se jednotlivé atributy podílí na konceptu (např. průměrná roční teplota je směrodatným atributem konceptu „ekosystémy v chladném podnebí“, kdežto průměrná roční vlhkost není směrodatným atributem tohoto konceptu). Formální koncept musí splňovat určitý vzájemný vztah mezi svým extentem a intentem, který zaručuje jeho maximálnost, jak dále uvidíme.

Budeme pracovat s konečným počtem objektů a atributů, které si pro jednoduchost očíslujeme. Označme množinu (indexů) objektů $X = \{1, \ldots, m\}$ a množinu (indexů) atributů $Y = \{1, \ldots, n\}$. Matice I spolu s množinami X a Y tvoří formální kontext. Dále označme symbolem V^U množinu všech zobrazení z množiny U do množiny V. Následují definice základních pojmů.

Definice 1.1. Fuzzy množina F nad univerzem U (v kontextu zvolené množiny stupňů L) je zobrazení $F : U \rightarrow L$.

Definice 1.2. Formální koncept $\langle C, D\rangle$ ve formálním kontextu $\langle X, Y, I\rangle$ je dvojice fuzzy množin $C \in L^X$ a $D \in L^Y$, které splňují

$$C = D^{\downarrow},\ D = C^{\uparrow},$$

kde $^{\uparrow} : L^X \to L^Y$ a $^{\downarrow} : L^Y \to L^X$ jsou zobrazení definována

$$C^{\uparrow}(y) = \bigwedge_{x \in X} (C(x) \to I_{xy}),$$
$$D^{\downarrow}(x) = \bigwedge_{y \in Y} (D(y) \to I_{xy}).$$

Jelikož jsou objekty a atributy očíslovány a tedy argumenty extentu a intentu jakožto fuzzy množin jsou indexy, lze si tyto množiny představit jako vektory (konečné posloupnosti) hodnot z L, tedy $C \in L^m$ a $D \in L^n$. Takto chápané extenty a intenty již tvoří rozklad matice I, jak dále uvidíme. Konkrétně pokud máme rozklad $I_{m\times n} = A_{m\times k} \circ B_{k\times n}$ sestávající z k faktorů (konceptů), pak extent l-tého konceptu tvoří l-tý sloupec matice A a jeho intent tvoří l-tý řádek matice B.

Pro kontext $\langle X, Y, I\rangle$ označme množinu všech konceptů $\mathcal{B}(X, Y, I)$. Pro danou podmnožinu konceptů $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ označme $A_{\mathcal{F}} \circ B_{\mathcal{F}}$ jako rozklad tvořený všemi koncepty z $\mathcal{F}$ (tedy $k = |\mathcal{F}|$). Budeme využívat následujících faktů prezentovaných v [4].

Věta 1.1. *Pro každou matici I existuje množina konceptů $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$, tak že platí $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$.*

Věta 1.2. *Pokud platí $I_{m\times n} = A_{m\times k} \circ B_{k\times n}$, pak existuje množina konceptů $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$, tak že platí $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$ a přitom $|\mathcal{F}| \leq k$.*

Formální koncepty jsou tedy vhodnými kandidáty na faktory. První věta říká, že pomocí formálních konceptů můžeme rozložit libovolnou matici. Druhá mluví o optimalitě formálních konceptů jakožto faktorů ve smyslu, že z nich lze vytvořit minimální rozklady.

Některé jiné přístupy k faktorové analýze tvoří rozklad z matic nad reálnými čísly (např. [2,8,10,11]) a k rekonstrukci původních dat používají klasický skalární součin reálných vektorů. Rozklady vytvořené těmito metodami mohou mít jinou doménu než vstupní data, například mohou obsahovat záporné hodnoty nebo desetinná čísla i přesto, že vstupní data byla nezáporná celá čísla. V takto vytvořených rozkladech se těžko interpretuje role faktorů a proces zpětné rekonstrukce dat. Role konceptů jakožto faktorů je přímočará a rozklad je přitom složen ze stejných stupňů jako v původních datech. Proces rekonstrukce původních dat tak lze dobře interpretovat. Stupeň, ve kterém objekt x nabývá původního atributu y je

$$I_{xy} = \bigvee_{\langle C,D\rangle \in \mathcal{F}} (C(x) \otimes D(y)),$$

kde vytvořeným rozkladem je $A_{\mathcal{F}} \circ B_{\mathcal{F}}$. Jde o složení původních dat z konceptů (operace $\bigvee$) a pro daný koncept se bere stupeň pravdivosti výroku „x je objektem konceptu a zároveň y je atributem konceptu“ (operace $\otimes$). Důležitá je zde volba operace $\otimes$, tak aby kombinace stupňů dávala intuitivní výsledky. Proto může tato metoda být zvlášť zajímavá v oblastech, kde jde zejména o nalezení nových znalostí v datech. Tato oblast je ale zatím málo probádaná.

1.2. Aplikace při kompresi obrazu

Z vlastností formálních konceptů plyne následující věta (uvedeno v [4]).

Věta 1.3. *Pro libovolnou množinu* $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ *platí* $A_{\mathcal{F}} \circ B_{\mathcal{F}} \leq I$.

Postupným přidáváním konceptů tedy rozklad aproximuje matici I zdola (maticové porovnání $\leq$ je myšleno po složkách). Samozřejmě v nejhorším případě z Věty 1.1. máme $A_{\mathcal{B}(X,Y,I)} \circ B_{\mathcal{B}(X,Y,I)} = I$. Typicky ale stačí ve srovnání s velikostí $\mathcal{B}(X, Y, I)$ malý počet konceptů pro vytvoření přibližného rozkladu $A_{\mathcal{F}} \circ B_{\mathcal{F}} \leq I$, který je matici I blízký. Navíc některé koncepty pokryjí větší část matice I než jiné a je tedy výhodné je z pohledu poměru přesnosti a velikosti rozkladu do rozkladu začlenit. Toho lze s výhodou využít pro minimalizaci rozkladu v oblastech, kde není požadovaná úplná přesnost rozkladu.

Jednou z možných aplikací je komprese obrazu. Na obrázek se lze dívat jako na trojici matic $\langle R, G, B\rangle$, které jsou tvořeny intenzitami pixelů obrázku v daném kanálu. Typicky jsou tedy matice tvořeny čísly v rozmezí 0 až 255. Můžeme uvažovat strukturu reziduovaného svazu $\langle\{0, \ldots, 255\}, \min, \max, \otimes, \rightarrow, 0, 255\rangle$. Operace $\otimes$ a $\rightarrow$ můžeme volit různě. Provedeme rozklad matic R, G a B a pokud jsou rozklady menší než původní matice, došlo ke kompresi obrazu. Aby byl rozklad o k faktorech menší než daný kanál obrázku o rozměrech $m \times n$, potřebujeme aby platilo $mk + kn < mn$, tedy pro počet faktorů musí platit $k < \frac{mn}{m+n}$.

Metodu lze snadno ovlivňovat parametry pro kvalitu rozkladu nebo pro velikost rozkladu. Můžeme požadovat určité procento přesně pokrytých pozic v matici, což ovlivní kvalitu rozkladu. Koncepty bychom pak hledali tak dlouho, dokud bychom nepokryli požadované množství pozic. Speciálním případem by pak byla hodnota 100 %, která by vedla na přesný rozklad. Druhým parametrem je počet faktorů k, který má přímý vliv na velikost rozkladu $k(m + n)$ (jednotek, typicky bajtů). Parametry lze také kombinovat.

Cílem této práce je prozkoumat potenciální využití rozkladu matic pomocí formálních konceptů pro kompresi obrazu.

1.3. Dosavadní experimenty

Metoda vznikla na Katedře informatiky Přírodovědecké fakulty Univerzity Palackého v Olomouci, kde byly také provedeny první experimenty. Tyto experimenty byly provedeny s malými množinami stupňů o velikosti okolo pěti stupňů. V rámci diplomové práce jsem vytvořil program, který implementuje faktorizační algoritmy a pracuje s obrázky a množinami stupňů $\{0, \ldots, n\}$ do velikosti 256. Experimenty jsem prováděl s množinou stupňů $\{0, \ldots, 255\}$, což je posun k mnohem většímu počtu stupňů.

2. Algoritmy

Algoritmus by měl prohledat prostor formálních konceptů $\mathcal{B}(X, Y, I)$ a vybrat z něj takovou podmnožinu $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$, aby platilo $A_{\mathcal{F}} \circ B_{\mathcal{F}} = I$ a počet konceptů $|\mathcal{F}|$ byl přitom co nejmenší. Bohužel [5] uvádí následující tvrzení pro speciální případ binárních dat.

Věta 2.1. *Problém nalezení rozkladu $I_{m\times n} = A_{m\times k} \circ B_{k\times n}$ pro co nejmenší k je NP-těžký.*

Bylo by totiž potřeba projít každou podmnožinu $\mathcal{B}(X, Y, I)$, kterých je exponenciálně mnoho vzhledem k velikosti $\mathcal{B}(X, Y, I)$. Přitom $\mathcal{B}(X, Y, I)$ může mít až exponenciální velikost vůči velikosti vstupních dat. Proto je potřeba využít hladových aproximačních algoritmů. Algoritmy budou iterativně prohledávat prostor konceptů a vyberou vždy jeden, který pokrývá nejvíce dosud nepokrytých pozic v matici I, a přidají ho do rozkladu. To budou opakovat až do limitní podmínky, kterou může být procento pokrytých pozic, maximální počet faktorů v rozkladu nebo jejich kombinace. Následuje popis dvou takových algoritmů.

2.1. Set Cover

Přímočaré řešení je využít hladové verze klasického algoritmu pro výpočet množinového pokrytí (set cover). Snažíme se pokrýt prvky matice I maticemi $C_{m\times 1} \circ D_{1\times n}$ pro daný koncept $\langle C, D\rangle \in \mathcal{F}$. Matice $C \circ D$ pokrývá prvek I_{ij} pokud $C(i) \otimes D(j) = I_{ij}$. Vypočítáme tedy nejprve celý konceptuální svaz $\mathcal{B}(X, Y, I)$ a poté z něj postupně přidáváme do rozkladu po jednom konceptu, vždy ten, který pokrývá nejvíce dosud nepokrytých pozic matice I.

Tímto způsobem sice není třeba procházet každou podmnožinu $\mathcal{B}(X, Y, I)$ a i když obecně přijdeme o optimální výsledek, většinou nebude vypočtený výsledek o moc horší, jelikož je vždy vybrán koncept co poryje nejvíc pozic, ale stále je potřeba opakovaně procházet celý konceptuální svaz a algoritmus bude velmi pomalý. Přitom nejsou přímo důležité samotné koncepty, ale jen jejich „pokrytí“, tedy boolovské matice $cover(C \circ D)$ velikosti $m \times n$ definované jako

$$(cover(C \circ D))_{i,j} = \begin{cases} 1 & \text{pokud } C(i) \otimes D(j) = I_{ij}, \\ 0 & \text{jinak.} \end{cases}$$

Mnoho různých konceptů bude mít stejné matice pokrytí. Proto je výhodné již při samotném počítání konceptuálního svazu hashovat matice $cover(C \circ D)$ a ukládat jen koncepty s unikátními maticemi pokrytí. Vypočteme tak podmnožinu konceptuálního svazu, ve které jsou všechny koncepty unikátní ve své matici pokrytí, a při vybírání konceptů do rozkladu tak opakovaně neprocházíme některé zbytečné koncepty. V sekci 5. je uvedeno do jaké míry tato optimalizace redukuje prohledávaný prostor.

Pro vygenerování konceptuálního svazu $\mathcal{B}(X, Y, I)$ používám algoritmus FCbO prezentovaný v [13]. Algoritmus 2.1. je upravenou verzí tohoto algoritmu, zobecněnou na libovolné lineárně uspořádané stupně a se zahrnutým filtrováním konceptů podle jejich matice pokrytí. Jedná se o rekurzivní algoritmus a počáteční volání je FastGenerateFrom($\langle \emptyset^{\downarrow}, \emptyset^{\downarrow\uparrow} \rangle$, 1, $\{\emptyset\}_{y \in Y}$). Spočítaný koncept $\langle A, B \rangle$ se uloží, pokud má mezi dosud uloženými koncepty unikátní matici pokrytí $cover(A \circ B)$. Poté se z konceptu $\langle A, B \rangle$ odvodí další koncepty. Argumenty y a posloupnost množin pro každý atribut $\{N_y\}_{y \in Y}$ plní optimalizační roli. Používají se k ořezání prohledávaného stromu, kdy není třeba sestoupit do větve, ve které by byly vygenerovány duplicitní koncepty. Výsledek algoritmu, tedy podmnožina konceptuálního svazu unikátní v maticích pokrytí, je postupně střádána na řádku 2 algoritmu 2.1. TOPELEMENT je nejvyšší stupeň (typicky 255). Pracuje se s fuzzy množinami. Definice průniku dvou fuzzy množin, definice fuzzy množin $Y_j : Y \to L$ a definice relace „je podmnožinou“ pro fuzzy množiny je následující:

$$\begin{aligned} (A \cap B)(x) &= A(x) \wedge B(x), \\ Y_j(y) &= \begin{cases} TOPELEMENT & \text{pokud } y < j, \\ 0 & \text{jinak,} \end{cases} \\ A \subseteq B &\iff \text{pro všechna } x \text{ platí } A(x) \leq B(x). \end{aligned}$$

Podrobnější popis včetně důkazu korektnosti je uveden v [13].

V algoritmu 2.2. nejprve pomocí algoritmu 2.1. naplníme množinu kandidátních konceptů $\mathcal{C} \subseteq \mathcal{B}(X, Y, I)$. Zbytek je pak již klasický algoritmus pro výpočet množinového pokrytí. V proměnné $\mathcal{U}$ se udržují dosud nepokryté pozice matice I. Limitní podmínka na řádku 4 algoritmu 2.2. může být upravena, tak aby se rozklad hledal jen do určité kvality (poměr pokrytých pozic), do maximálního počtu faktorů (velikost $\mathcal{F}$) nebo kombinace obojího. Z výsledné množiny faktorů $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ se může sestrojit rozklad.

2.2. Promising Columns

Velkým problémem algoritmu 2.2. je, že musí nejprve spočítat všechny koncepty (řádek 1 algoritmu 2.2.). Jelikož jich může být až exponenciálně mnoho vůči počtu atributů, tato samotná „předpříprava“ časově dominuje celý algoritmus. V článku [4] je uveden jiný algoritmus, který neprochází celý prostor konceptů. Snaží se hledaný koncept, co pokryje mnoho pozic, odhadnout pomocí tzv. „slibných sloupců“. Během toho sice spočítá mnoho dočasných konceptů, ale stále mnohem méně, než kdyby počítal celý svaz.

Nejprve uvádím definici sjednocení fuzzy množin a značení fuzzy singletonů

Algorithm 2.1. FastGenerateFrom($\langle A, B\rangle,\ y,\ \{N_y\}_{y\in Y}$)

```
1: if HasUniqueCover(A ∘ B) then
2:    store ⟨A, B⟩
3: end if
4: for j = y to n do
5:    M_j ← N_j
6:    if B(j) < TOPELEMENT and N_j ∩ Y_j ⊆ B ∩ Y_j then
7:       C ← A ∩ {^{B(j)+1}/_j}^↓
8:       D ← C^↑
9:       if B ∩ Y_j = D ∩ Y_j then
10:         put ⟨⟨C, D⟩, j⟩ to queue
11:      else
12:         M_j ← D
13:      end if
14:   end if
15: end for
16: while get ⟨⟨C, D⟩, j⟩ from queue do
17:   FastGenerateFrom(⟨C, D⟩, j, {M_y}_{y∈Y})
18: end while
19: return
```

Algorithm 2.2. SetCover(I)

```
1: 𝒞 ← FastGenerateFrom(⟨∅^↓, ∅^↓↑⟩, 1, {∅}_{y∈Y})
2: ℱ ← ∅
3: 𝒰 ← {⟨i, j⟩ | I_ij > 0}
4: while 𝒰 ≠ ∅ do
5:    select ⟨C, D⟩ ∈ 𝒞 that maximizes |𝒰 ∩ {⟨i, j⟩ | C(i) ⊗ D(j) = I_ij}|
6:    ℱ ← ℱ ∪ {⟨C, D⟩}
7:    𝒞 ← 𝒞 \ {⟨C, D⟩}
8:    𝒰 ← 𝒰 \ {⟨i, j⟩ | C(i) ⊗ D(j) = I_ij}
9: end while
10: return ℱ
```

(g je stupeň z L).

$$\begin{aligned}(A \cup B)(x) &= A(x) \vee B(x),\\ \{^g/_y\}(x) &= \begin{cases} g & \text{pokud } x = y,\\ 0 & \text{jinak.}\end{cases}\end{aligned}$$

Algoritmus promising columns je motivován následující vlastností intentů:

$$D = \bigcup_{y \in Y} \{^{D(y)}/_y\}^{\downarrow\uparrow}.$$

Tedy každý intent D se skládá z menších tzv. „sloupcových“ intentů $\{^{D(y)}/_y\}^{\downarrow\uparrow}$. U sloupcových intentů je kladen požadavek určitého stupně pouze pro jeden atribut (sloupec matice I). Ostatní atributy v něm mají nejvyšší možný stupeň (uzávěrový operátor $^{\downarrow\uparrow}$) co příslušný extent $\{^{D(y)}/_y\}^{\downarrow}$ dovolí. Algoritmus pak vychází z předpokladu, že dobrý intent (ten co spolu s příslušným extentem pokryje co nejvíc pozic matice I) se skládá z dobrých sloupcových intentů. Jelikož ale neznáme hledaný intent D, nemůžeme ani sestrojit jeho sloupcové intenty $\{^{D(y)}/_y\}^{\downarrow\uparrow}$ (závisí na D). Proto se k hledanému intentu postupně přibližujeme tak, že u dosavadního nejlepšího intentu zvýšíme stupeň v některém sloupci tak, aby po změně pokryl co nejvíce pozic. Jakoby se tím přiblížíme k jeho patřičnému sloupcovému intentu. Po změně stupně v sloupci je samozřejmě potřeba koncept přepočítat, tak aby to byl stále koncept.

Algoritmus 2.3. začne od intentu nejobecnějšího konceptu (řádek 4 algoritmu 2.3.) a snaží se ho maximalizovat opakovaným zvyšováním stupňů atributů (řádek 6 a 10 algoritmu 2.3.), tak aby nový koncept pokryl co nejvíce nových pozic

$$D \oplus {}^g/_y = \mathcal{U} \cap \{\langle i, j\rangle | (D \cup \{^g/_y\})^{\downarrow}(i) \otimes (D \cup \{^g/_y\})^{\downarrow\uparrow}(j) = I_{ij}\},$$

a to opakuje tak dlouho, dokud lze pokrýt více pozic (řádek 7 algoritmu 2.3.).

2.2.1. Implementační optimalizace

Časově nejnáročnější operací algoritmu je pro danou fuzzy množinu D výpočet extentu $D^{\downarrow}$ a uzávěru $D^{\downarrow\uparrow}$, který je potřeba provést v každé iteraci vnitřního cyklu na řádku 10 algoritmu 2.3. pro určení množiny $D \oplus {}^g/_y$, a to obecně pro každou dvojici atributu y a stupně g (množinu není nutné počítat pro dvojice $\langle y, g\rangle$ pro které platí $g \leq D(y)$, jelikož se pak $D \cup \{^g/_y\}$ neliší od D). Při výběru dvojice atributu a stupně na řádku 10 algoritmu 2.3. máme spočítaný průběžný intent $D = D^{\downarrow\uparrow}$ a extent $D^{\downarrow}$ (musel být spočítán pro určení $D \oplus {}^g/_y$ v některém předchozím kroku) a postupně přepočítáváme nový extent a intent po úpravě D na $D \cup \{^g/_y\}$ pro každou dvojici atributu a stupně. $D \cup \{^g/_y\}$ se od D liší pouze v bodě y, kde má nový vyšší stupeň g. Toho lze využít pro efektivnější výpočet $(D \cup \{^g/_y\})^{\downarrow}$ a $(D \cup \{^g/_y\})^{\downarrow\uparrow}$ ze znalosti $D^{\downarrow}$ a $D^{\downarrow\uparrow}$.

Algorithm 2.3. PromisingColumns(I)

$\mathcal{F} \leftarrow \emptyset$
$\mathcal{U} \leftarrow \{\langle i,j\rangle | I_{ij} > 0\}$
while $\mathcal{U} \neq \emptyset$ **do**
  $D \leftarrow \emptyset^{\downarrow\uparrow}$
  $V \leftarrow |\mathcal{U} \cap \{\langle i,j\rangle | \emptyset^{\downarrow}(i) \otimes \emptyset^{\downarrow\uparrow}(j) = I_{ij}\}|$
  select $\langle y, g\rangle$ that maximizes $|D \oplus {}^{g}/_{y}|$
  while $|D \oplus {}^{g}/_{y}| > V$ **do**
    $V \leftarrow |D \oplus {}^{g}/_{y}|$
    $D \leftarrow (D \cup \{{}^{g}/_{y}\})^{\downarrow\uparrow}$
    select $\langle y, g\rangle$ that maximizes $|D \oplus {}^{g}/_{y}|$
  end while
  $C \leftarrow D^{\downarrow}$
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{\langle C, D\rangle\}$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{\langle i,j\rangle | C(i) \otimes D(j) = I_{ij}\}$
end while
return $\mathcal{F}$

Nejprve se podíváme na $(D \cup \{{}^{g}/_{y'}\})^{\downarrow}$. Označme $D^{+} = D \cup \{{}^{g}/_{y'}\}$ a pro daný atribut y označme

$$\begin{aligned} r_{xy} &= D(y) \to I_{xy}, \\ r^{+}_{xy} &= D^{+}(y) \to I_{xy}. \end{aligned}$$

Potom pro $D^{\downarrow}$ a $D^{+\downarrow}$ z definice platí (pracujeme se strukturou reziduovaného svazu $\langle \{0,\ldots,255\}, \min, \max, \otimes, \to, 0, 255\rangle$)

$$\begin{aligned} D^{\downarrow}(x) &= \min_{y \in Y} r_{xy}, \\ D^{+\downarrow}(x) &= \min_{y \in Y} r^{+}_{xy}. \end{aligned}$$

Přitom pro dané x se r^{+}_{xy} liší od r_{xy} pouze pro $y = y'$ a to tak, že $r^{+}_{xy'} \leq r_{xy'}$ (operace $\to$ je antitonní v prvním argumentu a $D^{+}(y') > D(y')$). Platí tedy

$$D^{+\downarrow}(x) = \min(D^{\downarrow}(x), r^{+}_{xy'}). \tag{1}$$

Časovou složitost výpočtu $D^{+\downarrow}$ lze tedy snížit z $O(|X||Y|)$ na $O(|X|)$. Průměrně mi vyšlo zrychlení celého algoritmu promising columns asi o 20 %.

Pro zrychlení výpočtu $D^{+\downarrow\uparrow}$ nelze použít obdobný postup. Označme

$$\begin{aligned} r^{\downarrow}_{xy} &= D^{\downarrow}(x) \to I_{xy}, \\ r^{+\downarrow}_{xy} &= D^{+\downarrow}(x) \to I_{xy}. \end{aligned}$$

Potom pro $D^{\downarrow\uparrow}$ a $D^{+\downarrow\uparrow}$ z definice platí

$$\begin{aligned} D^{\downarrow\uparrow}(y) &= \min_{x\in X} r^{\downarrow}_{xy}, \\ D^{+\downarrow\uparrow}(y) &= \min_{x\in X} r^{+\downarrow}_{xy}. \end{aligned}$$

Ze vztahu (1) vidíme, že $D^{+\downarrow}$ může být oproti $D^{\downarrow}$ menší v libovolném bodě a tudíž se pro dané y mohlo zvýšit libovolné $r^{+\downarrow}_{xy}$ a je tedy nutné vztah přepočítat. Můžeme alespoň provést dílčí optimalizaci. Pro dané y si zapamatujeme tzv. definiční argument x_y, pro který platí $D^{\downarrow\uparrow}(y) = r^{\downarrow}_{x_y y}$ (tedy $x_y = \arg\min_{x\in X} r^{\downarrow}_{xy}$). Pokud se $r^{+\downarrow}_{x_y y}$ neliší od $r^{\downarrow}_{x_y y}$ (což znamená, že se $D^{+\downarrow}(x_y)$ neliší od $D^{\downarrow}(x_y)$), pak se ani $D^{+\downarrow\uparrow}(y)$ oproti $D^{\downarrow\uparrow}(y)$ nezměnilo (ostatní rezidua se mohla zvýšit, ale stále platí $\min_{x\in X} r^{+\downarrow}_{xy} = r^{\downarrow}_{x_y y}$). Jinak je nutné $D^{+\downarrow\uparrow}(y)$ přepočítat. $D^{+\downarrow\uparrow}$ tedy můžeme počítat podle vztahu

$$D^{+\downarrow\uparrow}(y) = \begin{cases} D^{\downarrow\uparrow}(y) & \text{pokud } D^{+\downarrow}(x_y) = D^{\downarrow}(x_y), \\ \min_{x\in X} D^{+\downarrow}(x) \to I_{xy} & \text{jinak.} \end{cases}$$

Asymptotická časová složitost výpočtu $D^{+\downarrow\uparrow}$ se tedy nemění a v praxi skutečně dochází většinou k přepočítávání. Průměrně mi vyšlo zrychlení celého algoritmu promising columns asi o 2 %. V jednom případě dokonce došlo k mírnému zpomalení kvůli režii zapamatovávání si x_y a testování, zda je nutné provést přepočet. Na rozdíl od předchozí optimalizace nemá tato dílčí optimalizace tedy moc význam.

Obě optimalizace ilustruje příklad 2.1.

▷ Příklad 2.1. Uvažujme množinu stupňů $L = \{0, 1, 2\}$, množinu objektů $X = \{x_1, x_2, x_3\}$ a množinu atributů $Y = \{y_1, y_2, y_3\}$. Pro jednodušší zápis značím v příkladu operaci min jako $\wedge$. Matice I a tabulka pro Łukasiewiczovo reziduum $\to$ na tříprvkovém řetězci jsou dány jako

$$I = \begin{pmatrix} 1 & 0 & 2 \\ 2 & 1 & 1 \\ 0 & 2 & 0 \end{pmatrix} \qquad \begin{array}{c|ccc} \to & 0 & 1 & 2 \\ \hline 0 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 2 & 0 & 1 & 2 \end{array}$$

Počáteční koncept $\langle D^{\downarrow}, D\rangle$ je $\langle \emptyset^{\downarrow}, \emptyset^{\downarrow\uparrow}\rangle = \langle \{{}^2/x_1, {}^2/x_2, {}^2/x_3\}, \{{}^0/y_1, {}^0/y_2, {}^0/y_3\}\rangle$. Definiční argumenty pro intent D jsou

$$x_{y_1} = x_3, \quad x_{y_2} = x_1, \quad x_{y_3} = x_3,$$

protože pro D platí (tučným písmem jsou označena aplikace rezidua, která určují výsledek jednotlivých $D(y)$)

$$\begin{aligned} D(y_1) &= (2 \to 1) \wedge (2 \to 2) \wedge (\mathbf{2} \to \mathbf{0}) = 1 \wedge 2 \wedge \mathbf{0} = \mathbf{0}, \\ D(y_2) &= (\mathbf{2} \to \mathbf{0}) \wedge (2 \to 1) \wedge (2 \to 2) = \mathbf{0} \wedge 1 \wedge 2 = \mathbf{0}, \\ D(y_3) &= (2 \to 2) \wedge (2 \to 1) \wedge (\mathbf{2} \to \mathbf{0}) = 2 \wedge 1 \wedge \mathbf{0} = \mathbf{0}. \end{aligned}$$

Nyní intent D rozšíříme na $D^+ = D \cup \{^1/y_1\}$. Následuje výpočet jednotlivých $D^{+\downarrow}(x)$ vždy nejprve klasickým způsobem a pak optimalizovaným způsobem (tučným písmem jsou označena aplikace rezidua použitá v optimalizovaném výpočtu).

$$\begin{aligned} D^{+\downarrow}(x_1) &= (\mathbf{1} \to \mathbf{1}) \wedge (0 \to 0) \wedge (0 \to 2) = \mathbf{2} \wedge 2 \wedge 2 = \mathbf{2}, \\ D^{+\downarrow}(x_1) &= 2 \wedge (\mathbf{1} \to \mathbf{1}) = 2 \wedge \mathbf{2} = \mathbf{2}, \end{aligned}$$

$$\begin{aligned} D^{+\downarrow}(x_2) &= (\mathbf{1} \to \mathbf{2}) \wedge (0 \to 1) \wedge (0 \to 1) = \mathbf{2} \wedge 2 \wedge 2 = \mathbf{2}, \\ D^{+\downarrow}(x_2) &= 2 \wedge (\mathbf{1} \to \mathbf{2}) = 2 \wedge \mathbf{2} = \mathbf{2}, \end{aligned}$$

$$\begin{aligned} D^{+\downarrow}(x_3) &= (\mathbf{1} \to \mathbf{0}) \wedge (0 \to 2) \wedge (0 \to 0) = \mathbf{1} \wedge 2 \wedge 2 = \mathbf{1}, \\ D^{+\downarrow}(x_3) &= 2 \wedge (\mathbf{1} \to \mathbf{0}) = 2 \wedge \mathbf{1} = \mathbf{1}. \end{aligned}$$

V optimalizované verzi výpočtu $D^{+\downarrow\uparrow}$ je nutné provést přepočet v bodech y_1 a y_3 a není nutné přepočítávat bod y_2 jelikož

$$D^{+\downarrow}(x_{y_1}) \neq D^{\downarrow}(x_{y_1}), \quad D^{+\downarrow}(x_{y_3}) \neq D^{\downarrow}(x_{y_3}), \quad D^{+\downarrow}(x_{y_2}) = D^{\downarrow}(x_{y_2}).$$

Následuje výpočet jednotlivých $D^{+\downarrow\uparrow}(y)$ ve stejném formátu jako výše. Pokud dojde k přepočtu, je potřeba pro budoucí výpočet aktualizovat patřičný definiční argument. Tučným písmem jsou označena aplikace rezidua, která určují výsledek jednotlivých $D^{+\downarrow\uparrow}(y)$.

$$\begin{aligned} D^{+\downarrow\uparrow}(y_1) &= (\mathbf{2} \to \mathbf{1}) \wedge (2 \to 2) \wedge (\mathbf{1} \to \mathbf{0}) = \mathbf{1} \wedge 2 \wedge \mathbf{1} = \mathbf{1}, \\ D^{+\downarrow\uparrow}(y_1) &= (\mathbf{2} \to \mathbf{1}) \wedge (2 \to 2) \wedge (\mathbf{1} \to \mathbf{0}) = \mathbf{1} \wedge 2 \wedge \mathbf{1} = \mathbf{1}, \end{aligned}$$

$$\begin{aligned} D^{+\downarrow\uparrow}(y_2) &= (\mathbf{2} \to \mathbf{0}) \wedge (2 \to 1) \wedge (1 \to 2) = \mathbf{0} \wedge 1 \wedge 2 = \mathbf{0}, \\ D^{+\downarrow\uparrow}(y_2) &= D(y_2) = \mathbf{0}, \end{aligned}$$

$$\begin{aligned} D^{+\downarrow\uparrow}(y_3) &= (2 \to 2) \wedge (\mathbf{2} \to \mathbf{1}) \wedge (\mathbf{1} \to \mathbf{0}) = 2 \wedge \mathbf{1} \wedge \mathbf{1} = \mathbf{1}, \\ D^{+\downarrow\uparrow}(y_3) &= (2 \to 2) \wedge (\mathbf{2} \to \mathbf{1}) \wedge (\mathbf{1} \to \mathbf{0}) = 2 \wedge \mathbf{1} \wedge \mathbf{1} = \mathbf{1}. \end{aligned}$$

Pokud je výsledek některého $D^{+\downarrow\uparrow}(y)$ určen více objekty, stačí za patřičný definiční argument zvolit libovolný z nich. Definiční argument x_{y_2} pro intent $D^{+\downarrow\uparrow}$ je stejný jako pro intent D, jelikož nedošlo k přepočtu. Definičními argumenty pro intent $D^{+\downarrow\uparrow}$ jsou tedy například

$$x_{y_1} = x_1, \quad x_{y_2} = x_1, \quad x_{y_3} = x_2.$$

□

3. Reprezentace reziduovaných svazů

Programu, který implementuje hledání rozkladů pomocí formálních konceptů, je potřeba nějakým způsobem zadat strukturu reziduovaného svazu. V úvodu bylo zmíněno, že pro hledání rozkladů obrázků lze uvažovat strukturu $\langle\{0,\ldots,255\},\min,\max,\otimes,\to,0,255\rangle$. Nejvyšším stupněm může samozřejmě být i jiné číslo než 255, pokud mají obrázky jinou bitovou hloubku než 8 bitů na každý kanál. Zbývá tedy zadat operace $\otimes$ a $\to$. Díky podmínce adjunkce reziduovaných svazů jedna z operací $\otimes$ nebo $\to$ jednoznačně určuje tu druhou, stačí tedy zadat jen jednu z nich, třeba operaci $\otimes$. Jednou z možností je uvést přímo jednu ze známých operací $\otimes$ (třeba minimum), ale to by výběr omezovalo jen na volbu jedné z několika operací $\otimes$ přímo implementovaných programem.

Operace násobení $\otimes$ reziduovaných svazů jsou tzv. zleva spojité t-normy (viz. [9] nebo definice 1.27 a 1.28 z [3], pojmy lze uvažovat na libovolném řetězci stupňů L, ne nutně pouze na reálném intervalu $[0,1]$). Spojité t-normy (viz. definice 1.36 z [3]) jsou speciálním případem zleva spojitých t-norem, které splňují tzv. zákon dělitelnosti $a\wedge b = a\otimes(a\to b)$. Pro spojité t-normy platí tzv. Mostert-Shieldsova reprezentace, která mluví o isomorfismu mezi reziduovanými svazy se spojitou t-normou a tzv. ordinálními součty. To umožní snazší reprezentaci operace $\otimes$. Nejprve ale definice ordinálního součtu (speciální verze definice 1.23 z [3]).

Definice 3.1. Mějme množinu indexů $I=\{0,1,\ldots,n\}$ pro $n\in N_0$ a systém $\{\mathbb{L}_i|i\in I\}$ úplných lineárních reziduovaných svazů $\mathbb{L}_i=\langle L_i,\wedge_i,\vee_i,\otimes_i,\to_i,0_i,1_i\rangle$, tak že pro $i=0,\ldots,n-1$ platí $1_i=0_{i+1}$ a $L_i\cap L_{i+1}=\{1_i\}$ a pro všechna $i,j\in I$, tak že $j>i+1$ platí $L_i\cap L_j=\emptyset$. Ordinální součet $\bigoplus_{i\in I}\mathbb{L}_i$ je úplný lineární reziduovaný svaz $\langle L,\wedge,\vee,\otimes,\to,0,1\rangle$, kde $L=\bigcup_{i\in I}L_i$, $0=0_0$, $1=1_n$ a $a\le b$ právě tehdy když buď $a,b\in L_i$ a $a\le_i b$, nebo $a\in L_i$, $b\in L_j$ a $i<j$. Operace $\wedge$ a $\vee$ jsou dány relací $\le$. operace $\otimes$ a $\to$ jsou definovány

$$a\otimes b = \begin{cases} a\otimes_i b & \text{pokud } a,b\in L_i,\\ a\wedge b & \text{jinak,}\end{cases}$$

$$a\to b = \begin{cases} 1 & \text{pokud } a\le b,\\ a\to_i b & \text{pokud } a\not\le b \text{ a } a,b\in L_i,\\ b & \text{jinak.}\end{cases}$$

Ordinální součet tedy spojí řetězce, které na sebe navazují, a počítá na nich tak, že pokud stupně patří do stejného podřetězce, je výsledek shodný s výsledkem aplikace odpovídající operace na tomto podřetězci (s výjimkou $a\to b$ pro případ $a\le b$, kde z adjunkce musí být výsledkem nejvyšší stupeň celého ordinálního součtu), jinak je výsledek stejný jako v Gödelově struktuře. Nyní již zmíněná reprezentace (verze věty 1.45 z [3]).

Věta 3.1. (Mostert-Shieldsova reprezentace) *Uvažujme spojitou t-normu $\otimes$ na konečném řetězci L. Pak úplný reziduovaný svaz odpovídající t-normě $\otimes$ je*

isomorfní s ordinálním součtem $\bigoplus_{i\in I} \mathbb{L}_i$ *úplných reziduovaných svazů* $\mathbb{L}_i$*, tak že* I *je množina všech idempotentů operace* $\otimes$ *a každé* $\mathbb{L}_i$ *je Łukasiewiczova algebra na intervalu* $[0_i, 1_i]$ *nebo jednoprvková algebra.*

Spojité t-normy $\otimes$ jsou tedy na konečných řetězcích jednoznačně dány svými idempotenty a lze je díky této reprezentaci zadat jejich výčtem. Idempotenti rozsekají stupně na podřetězce, tak že každý podřetězec je tvořen dvěma nejbližšími idempotenty, kterými je ohraničen, a stupni mezi nimi. Odpovídající reziduovaný svaz je pak isomorfní s ordinálním součtem těchto podřetězců. Lze tedy počítat přímo v tomto ordinálním součtu podle definice 3.1.

T-normy, které nejsou spojité, Mostert-Shieldsovu reprezentaci nesplňují. Například ji nesplňuje nilpotentní minimum, které je jen zleva spojité. Reprezentace nám však dává způsob, jak jednoduše zadat dostatečně bohatou třídu spojitých t-norem.

4. Použité nástroje

Jako součást diplomové práce jsem vytvořil konzolový program `gfd` (graded factor decomposition), který najde rozklad vstupního obrázku v zadané kvalitě pomocí algoritmu set cover nebo promising columns a uloží jej do souboru v binárním formátu. Z tohoto rozkladu je pak možné ve zpětném chodu programu `gfd` (viz. dokumentace programu v příloze A.) získat z něj vypočtený obrázek, popř. obrázky jednotlivých faktorů nebo textovou reprezentaci rozkladu. Pro práci s různými obrázkovými formáty využívá `gfd` knihovnu `CImg`, která externě volá program `convert` z balíčku `imagemagick`. Pro správný chod programu `gfd` je tedy nutné mít balíček `imagemagick` instalován (bývá instalován jako součást většiny linuxových distribucí).

Program `gfd` umožňuje zadat spojité t-normy pomocí výčtu jejich idempotentů (z nichž některé, jako třeba minimum, lze zadat jménem). Pro vygenerování delší posloupnosti idempotentů, které jsou od sebe stejně vzdálené, lze využít program `seq` (viz. dokumentace A.3.1.). Z t-norem, které jsou jen zleva spojité, program podporuje pouze nilpotentní minimum.

Jelikož je výpočet velmi časově náročný, byl mi zřízen přístup ke školnímu počítači, na kterém jsem vzdáleně spouštěl jednotlivé výpočty pomocí programu `screen`. Tímto způsobem jsem získal naprostou většinu výsledků.

Pro zmenšování obrázků, pro účel zkrácení doby výpočtu, jsem použil program `convert` (parametr `-resize`).

Pro sledování paměťové zátěže algoritmů jsem použil program `valgrind` a pro měření času běhu algoritmů jsem použil program `time`.

Grafy jsem vytvářel v programovacím jazyce `R`.

Metodu jsem testoval na obrázcích získaných z online zdrojů [1, 6, 12].

5. Výsledky

Pokud nebude uvedeno jinak, jsou všechny použité obrázky jednokanálové (šedotónové) nebo tříkanálové (barevné, model RGB) s 256 stupni.

5.1. Časová a paměťová efektivita algoritmů

Pokud nebude uvedeno jinak, jsou všechny uvedené časy naměřené na stroji Intel(R) Core(TM)2 Duo E8200 @ 2.66 GHz se 4 GiB RAM. Uvedené časy jsou tzv. user time, tedy čas strávený samotným výpočtem programu, sečteno přes všechna jádra. System time byl vždy zanedbatelný. Důvod, proč jsem zvolil user time místo real time je ten, že user time je odolný vůči vytíženosti stroje (k jednomu stroji jsem přistupoval vzdáleně a nebyl vyhrazen jen pro mě, nemohl jsem tedy kontrolovat vytíženost stroje) a vůči počtu jader (program může využít tolik jader, kolik je v obrázku kanálů, přitom jsem měl k dispozici stroje se dvěmi a čtyřmi jádry). User time je tedy jakýmsi ukazatelem vykonané práce. Časy lze převést na reálný čas podělením počtem kanálů obrázku nebo počtem jader procesoru, čehokoliv je méně. Časy jsou změřené pro celý běh programu, tedy včetně načtení obrázku a uložení rozkladu. Tato režie navíc (typicky se jednalo o 0.01 s) ovšem významně ovlivňuje pouze ta nejkratší měření, která jsou stejně problematicky měřitelná (časy se při opakovaných měřeních pohybovaly např. od 0.008 s do 0.020 s) a jejich celkový čas je zanedbatelný. Měření jsem několikrát opakoval, většinou třikrát, podle možností (neopakoval jsem například měření, které trvalo několik hodin) a bral jsem průměr časů. Až na ta zmíněná velmi krátká měření byla všechna měření velmi stabilní (většinou se jednotlivá měření lišila až na 4. platné pozici, málokdy na 3. platné pozici).

5.1.1. Set cover – vliv struktury

Vliv volby struktury na algoritmus set cover jsem otestoval na čtyřech šedotónových obrázcích velikosti 5×5 a jednom barevném obrázku velikosti 4×4 a nechal jsem provést úplné rozklady. Měl jsem tedy k dispozici 7 úplných rozkladů. Důvod, proč jsem volil tak absurdně malé obrázky je ten, že čas algoritmu rostl s velikostí obrázku velmi rychle. Například, když jsem zmenšil původní obrázek na velikost 4×4 a použil z hlediska rychlosti nejhorší strukturu, algoritmus trval 2 vteřiny. Když jsem stejný obrázek zmenšil na velikost 8×8 a použil stejnou strukturu, výpočet již trval 44 hodin. Velikost jsem tedy musel volit tak, aby algoritmus skončil v rozumné době i pro nejhorší strukturu.

Naměřené hodnoty pro jeden z těchto obrázků jsou uvedeny v tabulce 1. (ostatní obrázky vedly na obdobná pozorování, proto uvádím jen jeden). Za struktury jsem zvolil všechny struktury se spojitými operacemi $\otimes$, které mají pravidelně vzdálené idempotenty (step$X$ z tabulky 1. udává strukturu s operací $\otimes$, která má idempotenty násobky X, tedy 0, X, $2X$, …, 255), a strukturu s nil-

potentním minimem (pro jednoduchost označuji celou strukturu jako nilpotentní minimum). Tabulka uvádí pro danou strukturu počet idempotentů operace $\otimes$, počet konceptů, počet kandidátních konceptů (unikátních v matici pokrytí), počet faktorů rozkladu (algoritmem vybrané kandidátní koncepty) a čas běhu algoritmu ve vteřinách.

Tabulka 1. Porovnání vlivu struktury na algoritmus set cover. Pro jednotlivé struktury uvádí počty konceptů, kandidátních konceptů a faktorů a čas výpočtu ve vteřinách úplného rozkladu obrázku velikosti 5×5.

struktura	idempotentů	konceptů	kandidátů	faktorů	čas (s)
step1 (minimum)	256	222	150	5	0,015
step3	86	2 688	306	5	0,016
step5	52	5 333	293	5	0,027
step15	18	227 892	315	5	0,6
step17	16	173 155	347	5	0,45
step51	6	180 124	304	6	0,53
step85	4	10 002 065	394	6	26
step255 (Łukasiewicz)	2	12 196 422	325	6	37
nilpotentní minimum	129	131 989	280	5	0,37

Struktura má enormní vliv na velikost konceptuálního svazu. Optimalizace, která uchovává pouze koncepty unikátní v matici pokrytí, se jeví jako naprosto nezbytná. Víceméně bez ohledu na velikost konceptuálního svazu jsou počty kandidátních konceptů velmi podobné. V těch horších případech (vetší konceptuální svazy) tato optimalizace redukuje počet opakovaně procházených konceptů až o pět řádů. Z tohoto důvodu je čas algoritmu dominován fází průchodu konceptuálním svazem. Čas je víceméně přímo úměrný velikosti svazu.

Na první pohled by se mohlo zdát, že větší počet idempotentů vede na menší počet konceptů. Ve skutečnosti jde ale o vhodné idempotenty pro daný obrázek. Vetší počet idempotentů pouze pravděpodobněji obsahuje vhodné idempotenty, což je patrné ze dvou případů. Prvním případem je nilpotentní minimum, které si nevedlo moc dobře, i přes svůj velký počet idempotentů. Nilpotentní minimum má idempotenty 0 a pak všechny stupně od 128 nahoru. Tato nepravidelná distribuce může za slabší výkon, což potvrdil i test se třemi idempotenty, kdy jsem za idempotenty zvolil povinné 0 a 255 a s třetím jsem hýbal. Když jsem jej volil blízko 0 nebo 255, byl výsledek podobný jako pro Łukasiewiczovo $\otimes$, když jsem se s ním blížil ke středu stupňů, výsledek se lepšil. Druhým případem je step15, který si vedl hůře než step17 a step51. Step17 a step51 obsahovaly pro tento obrázek vhodnější idempotenty. Tato pozorování se potvrdila i pro zbylých šest rozkladů. Zejména step15 a step17 se často střídaly v počtu konceptů, jelikož mají téměř stejný počet idempotentů, jde spíše o vhodnost idempotentů pro daný obrázek. Step15 měl méně konceptů ve čtyřech ze sedmi případů, dva nejextrémnější poměry počtů konceptů byly 1:3 ve prospěch step17 a 2:5 ve prospěch step15. Nil-

potentní minimum se pohybovalo co do počtu konceptů mezi 4. a 7. místem. Minimum (Gödel) vyšlo vždy nejlépe a Łukasiewicz vyšel vždy nejhůře, což opět posiluje argument vhodných idempotentů, jelikož minimum má všechny idempotenty a nemůže tedy být nikdy horší než jiná operace $\otimes$ a naopak všechny ostatní operace $\otimes$ mají alespoň idempotenty 0 a 255 a nemohou tedy být horší než Łukasiewicz.

Počty kandidátních konceptů se mezi různými strukturami s výjimkou minima, které mělo téměř vždy méně samotných konceptů než ostatní struktury kandidátních konceptů, moc nelišily a byly chaotické (nebyl patrný vliv počtu idempotentů).

Z tabulky 1. by se mohlo zdát, že více idempotentů vede na méně faktorů, ale to se u ostatních rozkladů nepotvrdilo. Počty faktorů pro všechny rozklady jsou uvedeny v tabulce 2.

Tabulka 2. Počty faktorů úplných rozkladů několika obrázků velikosti 5×5 vypočtených algoritmem set cover pro jednotlivé struktury.

struktura	obr. 1	obr. 2	obr. 3	obr. 4	obr. 5R	obr. 5G	obr. 5B
step1 (minimum)	6	5	6	6	5	4	4
step3	6	5	6	6	5	4	4
step5	6	5	6	6	5	4	4
step15	5	5	5	6	5	4	4
step17	6	5	6	6	4	4	4
step51	6	6	6	6	5	5	4
step85	6	6	5	6	5	5	6
step255 (Łukas.)	6	6	6	6	5	5	5
nilpotentní min.	8	5	6	6	4	4	5

Počty konceptů mezi různými obrázky se také lišily výrazněji pro horší struktury. Z obrázků velikosti 5×5 měl pro Łukasiewiczovu strukturu nejjednodušší obrázek 403 629 konceptů a nejsložitější 35 372 467 konceptů. Pro minimum měl nejjednodušší obrázek 222 konceptů a nejsložitější 252 konceptů.

Nejlepší volbou struktury se tedy jeví minimum. Minimum vede na nejmenší počty konceptů a kandidátních konceptů, což významně zrychluje algoritmus a přitom nevede na nijak horší rozklady. Minimum je také nejstabilnější pro různé obrázky, jelikož má všechny idempotenty.

5.1.2. Promising columns – vliv struktury

Algoritmus promising columns jsem otestoval na 57 obrázcích převážně velikosti 100×100 (8 z nich mělo velikost 128×128 a jeden byl 100×67), z toho 24 bylo barevných. Měl jsem tedy k dispozici celkem 105 úplných rozkladů pro každou strukturu. Časy byly změřené na stroji Intel(R) Core(TM) i7-2600 @ 3.40 GHz se 12 GiB RAM. Volba struktury se u tohoto algoritmu nejevila významná, výsledky vycházely velmi podobně pro všechny struktury. Pro představu

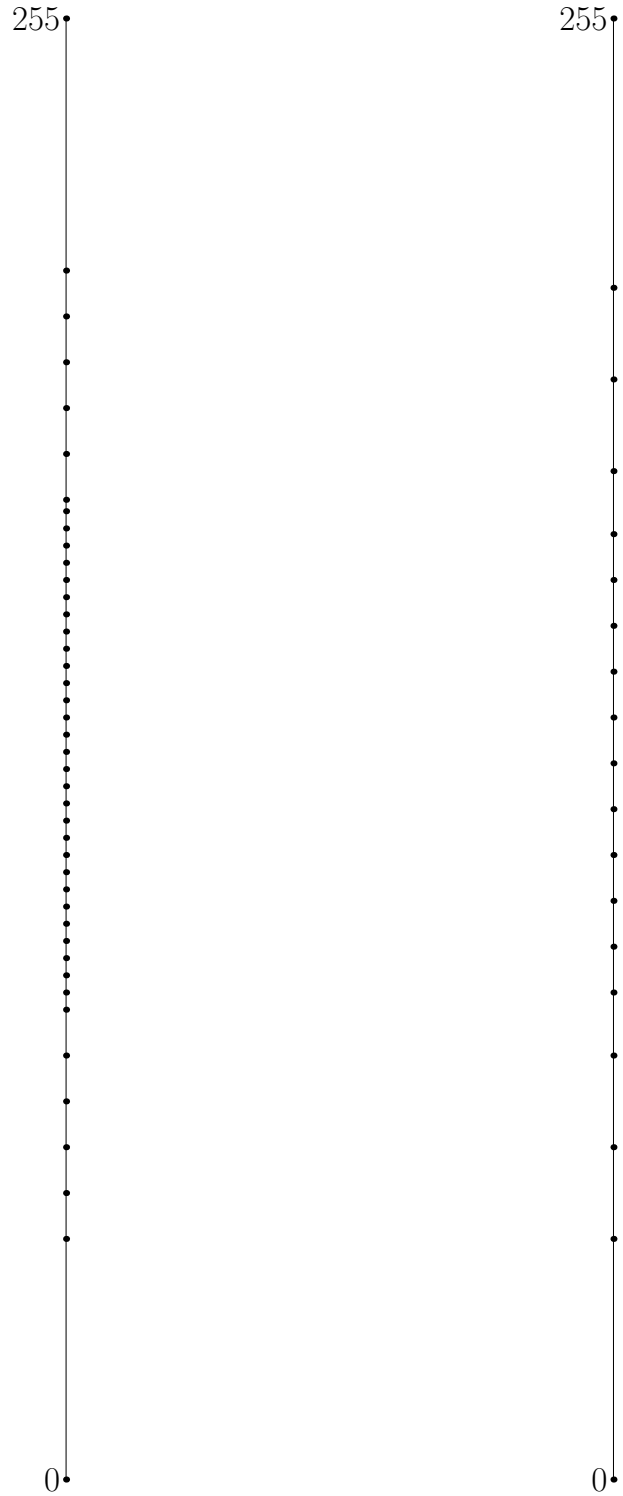
uvádím v tabulce 3. údaje pro jeden náhodně vylosovaný barevný obrázek velikosti 100×100. Tabulka pro každou volbu struktury uvádí počet idempotentů operace $\otimes$, počet faktorů úplného rozkladu v každém kanále a dobu výpočtu v minutách.

Tabulka 3. Počty faktorů rozkladů jednotlivých kanálů obrázku velikosti 100×100 vypočtených algoritmem promising columns pro jednotlivé struktury a celkový čas výpočtu v minutách.

struktura	idempotentů	kanál R	kanál G	kanál B	čas (m)
step1 (minimum)	256	136	131	117	13,9
step3	86	131	130	116	13,9
step5	52	134	125	124	14,4
step15	18	127	123	126	14,5
step17	16	124	122	118	13,9
step51	6	136	124	124	14,4
step85	4	133	128	115	13,9
step255 (Łukasiewicz)	2	140	142	133	13,8
nilpotentní minimum	129	137	132	148	13,4
bottomfew	9	145	141	120	14,4
densemid	43	128	126	119	14,4
sparsemid	19	129	122	122	14,5

Bottomfew je struktura se spojitou operací $\otimes$, která má idempotenty 0, 2, 4, 6, 16, 18, 52, 86 a 255. Jde tedy o protiklad nilpotentního minima s pár idempotenty v nízkých stupních. Densemid a sparsemid jsou struktury se spojitými $\otimes$, pro které jsem idempotenty koncentroval do středu stupňů. Stupně jsem rozdělil do šestin a v každé šestině jsem zvolil idempotenty v pravidelném kroku v odlišné hustotě. Ve spodní a horní šestině jsem nechal pouze povinné 0 a 255. V prostředních dvou šestinách jsem zvolil více idempotentů a ve 2. a 5. šestině jsem zvolil méně idempotentů. Konkrétně pro densemid jsem zvolil krok pro 2. a 5. šestinu 8 a pro prostřední šestiny krok 3. Pro sparsemid jsem zvolil krok pro 2. a 5. šestinu 16 a pro prostřední šestiny krok 8. Rozmístění idempotentů pro tyto dvě struktury jsou znázorněny na obrázku 1.

Jelikož tedy není na první pohled vliv struktury příliš patrný, provedl jsem porovnání jinak. Pro každý obrázek jsem vypočítal průměr časů přes všechny struktury a jednotlivé časy struktur jsem vyjádřil jako poměr vůči tomuto průměru. Tedy například, pokud byl průměrný čas pro jeden obrázek 10 minut a minimum mělo čas 11 minut, dostal jsem poměr $\frac{11}{10}$, což vyjadřuje, že na tomto obrázku si minimum vedlo o 10 % hůře než průměrná struktura. Tímto způsobem jsem pro každou strukturu dostal 57 poměrů (pro každý obrázek jeden), ze kterých jsem spočítal průměr. Výsledky ukazuje tabulka 4., která pro danou strukturu uvádí průměr poměrů počtu faktorů (získáno obdobně jako pro čas, akorát jsem poměr vzal pro každý kanál, dohromady tedy jde o průměr 105 poměrů), průměr



(a) densemid (b) sparsemid

Obrázek 1. Rozmístění idempotentů struktur densemid a sparsemid

časových poměrů, směrodatnou odchylku časových poměrů a nakonec nejlepší a nejhorší časový poměr.

Tabulka 4. Porovnání vlivu struktury na algoritmus promising columns. V tabulce jsou uvedeny poměry měřených atributů mezi strukturami. Menší poměr znamená lepší výsledek. Pro jednotlivé struktury je uveden průměr poměrů počtu faktorů, průměr časových poměrů, směrodatná odchylka časových poměrů a nejlepší a nejhorší časový poměr.

struktura	průměr f.	průměr č.	sm. od.	min	max
step1 (minimum)	0,992	1,055	0,060	0,867	1,187
step3	0,992	1,060	0,064	0,866	1,188
step5	0,998	1,057	0,060	0,868	1,204
step15	0,997	1,038	0,055	0,874	1,133
step17	0,998	1,030	0,057	0,875	1,191
step51	0,993	0,984	0,048	0,805	1,067
step85	0,999	0,978	0,076	0,887	1,428
step255 (Łukasiewicz)	1,024	0,866	0,181	0,486	1,658
nilpotentní minimum	1,011	0,873	0,165	0,527	1,549
bottomfew	1,009	1,013	0,080	0,836	1,453
densemid	0,994	1,028	0,047	0,880	1,133
sparsemid	0,994	1,020	0,048	0,879	1,143

S výjimkou Łukasiewiczovi struktury a nilpotentního minima vyšly struktury velmi podobně. Rozdíly byly v řádu procent jak u průměrů, tak v odchylkách. Mezi nimi by se snad dala vyzdvihnout struktura step51, která měla dobrou kombinaci všech sledovaných atributů. Vedla na jedny z nejmenších rozkladů, přitom byla jedna z nejrychlejších, měla jednu z nejmenších odchylek, 3. nejlepší nejlepší případ a vůbec nejlepší nejhorší případ. Rozdíly byly ovšem velmi malé. Naopak Łukasiewiczova struktura a nilpotentní minimum se od ostatních docela lišily, bohužel v některých ohledech směrem k horšímu. Měly o 1–3 % horší rozklady, ale zato byly o 10–20 % rychlejší. Na druhou stranu byly zase méně stabilní se skoro trojnásobnou odchylkou.

Z hlediska vlivu na dobu výpočtu tedy není volba struktury pro algoritmus promising columns kritická. Doporučit lze Łukasiewiczovu strukturu a nilpotentní minimum, které přece jen byly průměrně o něco rychlejší, i když s velkými výkyvy.

5.1.3. Vzájemné srovnání algoritmů

Vzhledem k výše uvedeným pozorováním jsem pro vzájemné srovnání obou algoritmů zvolil strukturu minimum. Dalo se čekat, že si set cover povede hůř, takže minimum, které je pro něj nejlepší volba a zároveň nepatří k těm rychlejším pro promising columns, dává set coveru největší šanci. Porovnání jsem provedl na čtyřech šedotónových obrázcích velikosti 10×10. Výsledky jsou v tabulce 5.

Jde o úplné rozklady. Tabulka uvádí čas výpočtu pro oba algoritmy ve vteřinách a počet faktorů rozkladu.

Tabulka 5. Vzájemné srovnání algoritmů set cover a promising columns. Pro jednotlivé obrázky jsou uvedeny počty faktorů a čas výpočtu ve vteřinách úplných rozkladů vypočtených algoritmy set cover a promising columns.

	SC faktorů	PC faktorů	čas SC (s)	čas PC (s)
obr. 1	12	14	0,85	0,086
obr. 2	15	11	1,7	0,066
obr. 3	16	14	1,6	0,1
obr. 4	15	13	1,5	0,064

Tabulka hovoří jasně, promising columns je řádově rychlejší, přitom dokonce vede na o něco lepší rozklady, což mě překvapilo. Doba výpočtu set coveru navíc roste s velikostí vstupu mnohem rychleji. Pro obrázek velikosti 16×16 vzrostl čas pro promising columns na 0.17 s, zatímco čas set coveru vzrostl na 5700 s. Cena za počítání celého konceptuálního svazu je tedy příliš vysoká, přitom se ani nakonec neprojeví v kvalitě rozkladu.

5.1.4. Paměťová náročnost

Promising columns není paměťově náročný. Kromě místa pro samotné faktory potřebuje akorát místo pro tři dočasné koncepty, se kterými aktuálně počítá (aktuální kandidát na faktor, jeho aktuální nejlepší rozšíření a právě ověřovaný koncept při hledání lepšího rozšíření). Zato set cover je velmi paměťově náročný. Musí uložit alespoň všechny kandidátní koncepty, ale zejména dochází k prohledávání prostoru konceptů, během kterého se z aktuálního konceptu odvodí mnoho dalších, které je potřeba dočasně uložit, než budou zpracovány. Při velikosti konceptuálních svazů to je další velká nevýhoda set coveru.

Například pro obrázek velikosti 10×10 vedl program s použitím algoritmu promising columns celkem na 114 alokací a dohromady alokoval asi 160 KiB. S použitím set cover algoritmu to bylo asi 3 milióny alokací a dohromady asi 180 MiB. Pro obrázek velikosti 119×8124 nepřesáhlo pro promising columns využití paměti 2.2 MiB. Pro stejný obrázek a set cover dosáhlo využití paměti 28 GiB (12 GiB RAM a 16 GiB swap), poté byl výpočet přerušen.

5.1.5. Shrnutí efektivity algoritmů

Vzhledem k výše uvedeným pozorováním je jasným favoritem algoritmus promising columns. Je výrazně časově i paměťově efektivnější, přitom vede minimálně na stejně kvalitní rozklady, spíše i o něco lepší. Z tohoto důvodu se ve zbytku textu zabývám jen algoritmem promising columns.

5.2. Kvalita rozkladů

Následující měření jsem provedl na 57 obrázcích převážně velikosti 100×100 (8 z nich mělo velikost 128×128 a jeden byl 100×67), z toho 24 bylo barevných. Měl jsem tedy k dispozici celkem 105 úplných rozkladů pro každou strukturu.

5.2.1. Úplné rozklady

Tabulka 6. obsahuje data pro úplné rozklady. Opět jsem použil poměry pro každý rozklad (kanál obrázku). Pro každý rozklad jsem si tedy vyjádřil velikost rozkladu pro danou strukturu jako poměr

$$\frac{\text{počet faktorů pro danou strukturu}}{\text{průměrný počet faktorů přes všechny struktury}}.$$

Z těchto poměrů jsem pak sestavil tabulku, která pro danou strukturu uvádí počet idempotentů operace $\otimes$, průměr poměrů, směrodatnou odchylku poměrů a nejlepší a nejhorší poměr.

Tabulka 6. Vliv struktury na úplné rozklady. Tabulka vychází z poměrů počtu faktorů mezi strukturami pro jednotlivé rozklady. Menší poměr znamená menší rozklad. Pro jednotlivé struktury je uveden průměr poměrů počtu faktorů, směrodatná odchylka poměrů a nejlepší a nejhorší poměr.

struktura	idempotentů	průměr	sm. od.	min	max
step1 (minimum)	256	0,992	0,029	0,890	1,059
step3	86	0,992	0,032	0,920	1,070
step5	52	0,998	0,031	0,924	1,148
step15	18	0,997	0,027	0,932	1,065
step17	16	0,998	0,032	0,910	1,103
step51	6	0,993	0,033	0,895	1,059
step85	4	0,999	0,037	0,898	1,113
step255 (Łukasiewicz)	2	1,024	0,059	0,706	1,162
nilpotentní minimum	129	1,011	0,098	0,681	1,276
bottomfew	9	1,009	0,038	0,909	1,094
densemid	43	0,994	0,025	0,917	1,059
sparsemid	19	0,994	0,029	0,910	1,063

Pro úplné rozklady je volba struktury zanedbatelná. Snad jen Łukasiewiczova struktura je mírně horší. Méně stabilní je Łukasiewiczova struktura a zejména pak nilpotentní minimum, u kterého směrodatná odchylka tvoří téměř 10 %.

5.2.2. Částečné rozklady

Kvalitu částečných rozkladů jsem měřil pomocí dvou metrik. Jednou byl poměr pokrytých pozic a druhou byla blízkost rekonstruované matice k matici pů-

vodní. Pro určení blízkosti jsem vycházel z průměrné vzdálenosti *avgdist* jednotlivých hodnot zrekonstruované matice k hodnotám původní matice. Konkrétně jsem blízkost kvantifikoval procentuálně podle vztahu

$$\left(1 - \frac{avgdist}{255}\right) \cdot 100.$$

Poměr pokrytých pozic dává garanci nějakého počtu přesných hodnot, ale vůbec nebere v potaz vzdálenost na ostatních pozicích. Z tohoto pohledu se blízkost jeví jako vhodnější metrika. Výše definovaná blízkost je ovšem poněkud zavádějící. Typicky již prvních pár faktorů splňuje 80 % kvalitu. Typické obrázky mají většinu hodnot kolem středu a 80 % kvalita znamená, že jsou hodnoty průměrně vzdálené asi o 50 stupňů, tedy například místo hodnoty 130 rozklad reprezentuje hodnotu 80, což je samozřejmě velmi nekvalitní. Pro blízkost je tedy nutné sledovat mnohem vyšší kvality.

Pro poměr pokrytých pozic jsem zvolil krok 10 %. Měření jsou uvedena v tabulce 7. Až na Łukasiewiczovu strukturu není vliv struktury významný. Łukasiewiczova struktura má zajímavý průběh. Zpočátku si vede průměrně, pak si ale dlouho udržuje v porovnání s ostatními strukturami zhruba o 10 % menší rozklady. Bohužel pro z praktického pohledu zajímavé kvality (80 % a víc) se opět vrací k průměru a nakonec v úplném rozkladu je dokonce nejhorší strukturou, jak již bylo uvedeno. Nilpotentní minimum má zpočátku velké rozklady, později se však dostává na druhé místo.

Tabulka 7. Vliv struktury na velikost částečných rozkladů splňujících dané procento pokrytých pozic. Tabulka vychází z poměrů počtu faktorů mezi strukturami pro jednotlivé rozklady. Menší poměr znamená menší rozklad. Pro jednotlivé struktury a kvality je uveden průměr poměrů počtu faktorů nejmenších rozkladů splňující danou kvalitu.

struktura	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %	100 %
minimum	0,98	1,02	1,03	1,02	1,03	1,02	1,02	1,01	1,00	0,99
step3	0,98	1,03	1,03	1,02	1,02	1,02	1,02	1,01	1,00	0,99
step5	1,00	1,02	1,02	1,02	1,02	1,02	1,02	1,01	1,01	1,00
step15	1,01	1,01	1,02	1,02	1,02	1,02	1,02	1,01	1,01	1,00
step17	1,00	1,01	1,01	1,02	1,02	1,02	1,02	1,01	1,01	1,00
step51	0,99	0,99	1,00	1,00	1,01	1,01	1,01	1,01	1,01	0,99
step85	0,98	0,97	0,97	0,97	0,98	0,99	0,99	1,00	1,00	1,00
Łukas.	0,99	0,92	0,91	0,89	0,89	0,90	0,91	0,94	0,97	1,02
nilp. min.	1,11	1,03	0,99	0,98	0,96	0,96	0,97	0,97	0,98	1,01
bottomfew	0,97	1,01	1,01	1,03	1,01	1,01	1,00	1,00	1,00	1,01
densemid	0,98	1,00	1,01	1,01	1,02	1,02	1,01	1,01	1,01	0,99
sparsemid	1,00	0,99	1,00	1,01	1,01	1,02	1,02	1,01	1,01	0,99

Pro blízkost jsem zvolil z výše uvedeného důvodu vysoké kvality s krokem 2 %. Měření jsou uvedena v tabulce 8. Chování struktur je obdobné jako v poměru pokrytých pozic, ale mnohem výraznější. Łukasiewiczova struktura má ve středních

kvalitách až o čtvrtinu menší rozklady. Kvalita 96 %, pro kterou si Łukasiewiczova struktura stále udržuje dominantní postavení, znamená průměrně asi o 10 stupňů menší hodnoty (rozklady aproximují původní obrázek zdola). Nilpotentní minimum si opět vede velmi špatně v nízkých kvalitách, ale mnohem lépe později.

Tabulka 8. Vliv struktury na velikost částečných rozkladů splňujících danou blízkost. Tabulka vychází z poměrů počtu faktorů mezi strukturami pro jednotlivé rozklady. Menší poměr znamená menší rozklad. Pro jednotlivé struktury a kvality je uveden průměr poměrů počtu faktorů nejmenších rozkladů splňující danou kvalitu.

struktura	82 %	84 %	86 %	88 %	90 %	92 %	94 %	96 %	98 %	100 %
minimum	1,02	1,03	1,03	1,03	1,04	1,06	1,06	1,05	1,04	0,99
step3	1,00	1,00	1,01	1,03	1,04	1,04	1,05	1,05	1,04	0,99
step5	1,02	1,02	1,01	1,03	1,04	1,05	1,05	1,05	1,04	1,00
step15	0,99	1,00	1,01	1,03	1,03	1,03	1,04	1,04	1,03	1,00
step17	1,00	1,00	1,00	1,02	1,04	1,04	1,06	1,05	1,03	1,00
step51	0,99	1,00	1,00	1,01	1,02	1,02	1,03	1,04	1,02	0,99
step85	0,96	0,96	0,99	0,98	0,97	0,97	0,97	0,97	1,00	1,00
Łukas.	0,81	0,80	0,78	0,76	0,76	0,77	0,75	0,76	0,83	1,02
nilp. min.	1,22	1,19	1,16	1,10	1,03	0,96	0,91	0,90	0,92	1,01
bottomfew	0,97	0,97	0,97	0,97	0,98	0,97	0,97	0,98	0,98	1,01
densemid	1,01	1,01	1,02	1,01	1,02	1,05	1,06	1,06	1,04	0,99
sparsemid	1,01	1,03	1,02	1,03	1,05	1,05	1,05	1,05	1,03	0,99

Celkově lze pro částečné rozklady určitě doporučit Łukasiewiczovu strukturu, obzvlášť pokud není požadavek na kvalitu vysoký. Díky menším rozkladům bude výpočet i rychlejší. Je ovšem důležité rozlišovat mezi částečnými rozklady a úplnými. Łukasiewiczova struktura má nejmenší částečné rozklady, ale přitom největší úplné rozklady. Podobně nilpotentní minimum má druhé nejmenší částečné rozklady (až na hodně nízké kvality), ale druhé nejhorší úplné rozklady. Naopak třeba minimum patřilo v částečných rozkladech k těm nejhorším, ale dopadlo mezi nejlepšími v úplných rozkladech.

5.2.3. Průběh rozkladu

Průběh rozkladu jsem měřil na podmnožině obrázků velikosti 100×100 (48 obrázků, 80 rozkladů). Měření jsem provedl ze dvou pohledů. Prvním pohledem byl průměrný počet faktorů pro danou kvalitu a druhým byl opačný pohled, tedy dosažená kvalita pro daný počet faktorů.

Naměřená data z pohledu kvality jsou uvedena v tabulce 9. pro pokryté pozice a v tabulce 10. pro blízkost. Tabulky udávají pro danou strukturu průměrný počet faktorů, který byl potřeba pro dosažení dané kvality. Na obrázku 2. jsou pro tři vybrané struktury naměřená data proložena křivkou. V datech je patrný charakter algoritmu. Zaprvé algoritmus postupně hladově vybírá koncepty, co pokryjí nejvíc pozic, a zadruhé postupným přidáváním konceptů do rozkladu klesá

počet nepokrytých pozic a tím klesá i počet nově pokrytých pozic s přidáváním dalších konceptů. Kvalita tedy ze začátku roste rychle, ale čím je rozklad větší, tím efektivita přidávání dalších konceptů klesá. V závěru je přidávání konceptů velmi neefektivní, kdy je třeba velikost rozkladu téměř zdvojnásobit, aby bylo pokryto posledních 10 % pozic.

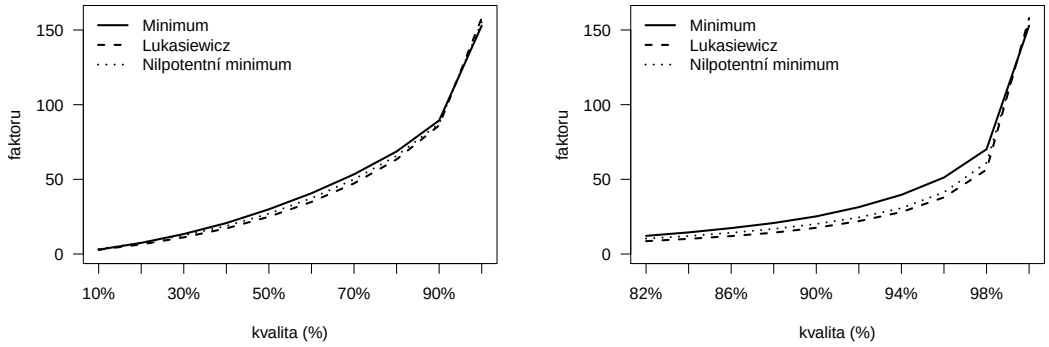
Tabulka 9. Průběh velikosti rozkladů z pohledu procenta pokrytých pozic. Pro jednotlivé struktury a kvality uvádí kolik bylo průměrně potřeba faktorů v rozkladu pro splnění dané kvality.

struktura	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %	100 %
minimum	3	7	13	21	30	41	53	69	89	153
step3	3	8	13	21	30	41	53	69	89	153
step5	3	7	13	21	30	41	53	69	89	153
step15	3	7	13	21	30	41	53	69	90	154
step17	3	7	13	21	30	41	53	69	89	153
step51	3	7	13	20	29	40	53	68	90	153
step85	3	7	12	19	29	39	52	68	89	153
Łukas.	3	6	11	17	25	35	47	63	86	158
nilp. min.	3	7	13	19	27	37	50	66	87	155
bottomfew	3	7	13	21	30	40	53	68	90	154
densemid	3	7	13	20	29	40	53	69	90	153
sparsemid	3	7	13	20	29	40	53	69	89	153

Tabulka 10. Průběh velikosti rozkladů z pohledu blízkosti. Pro jednotlivé struktury a kvality uvádí kolik bylo průměrně potřeba faktorů v rozkladu pro splnění dané kvality.

struktura	82 %	84 %	86 %	88 %	90 %	92 %	94 %	96 %	98 %	100 %
minimum	12	14	17	21	25	31	40	51	70	153
step3	12	14	17	21	25	31	40	51	70	153
step5	12	14	17	21	25	31	39	51	70	153
step15	12	14	17	21	25	31	39	51	70	154
step17	12	14	17	21	25	31	40	51	70	153
step51	12	14	17	20	25	30	39	50	69	153
step85	12	14	17	20	24	30	37	49	68	153
Łukas.	9	10	12	14	18	22	28	38	56	158
nilp. min.	10	12	14	17	20	24	31	41	61	155
bottomfew	12	14	17	20	24	30	37	49	67	154
densemid	12	15	17	21	25	31	40	51	70	153
sparsemid	13	15	18	21	26	32	40	51	70	153

Naměřená data z pohledu počtu faktorů jsou uvedena v tabulkách 11. a 12. Proložení dat přímkou pro vybrané struktury je na obrázku 3. Tato data ukazují jaké kvality je dosaženo pro různé míry komprese. Velikost rozkladu je stejná jako



(a) Pokryté pozice (b) Blízkost

Obrázek 2. Průběh rozkladu z pohledu kvality

velikost původního obrázku pro $k = \frac{mn}{m+n}$, kde k je počet faktorů a m a n jsou rozměry obrázku. Pro obrázky velikosti 100×100, na kterých byla data naměřena, dochází tedy k rovnosti velikosti rozkladu a původního obrázku na 50 faktorech. Z tabulek lze vyčíst, že když necháme rozklad narůst až na velikost původního obrázku, dostaneme kvalitu zhruba 70 % pokrytých pozic nebo 95 % blízkosti. I při použití 100 faktorů a tedy dvojnásobné velikosti rozkladu vůči velikosti původního obrázku, dostaneme stále jen asi 90 % pokrytých pozic. To z hlediska možného využití metody pro komprimaci obrazu není dobrá zpráva. Pokud bychom chtěli komprimovat alespoň na polovinu původní velikosti (25 faktorů), pak se musíme spokojit s kvalitou kolem 50 % pokrytých pozic nebo 90 % blízkosti.

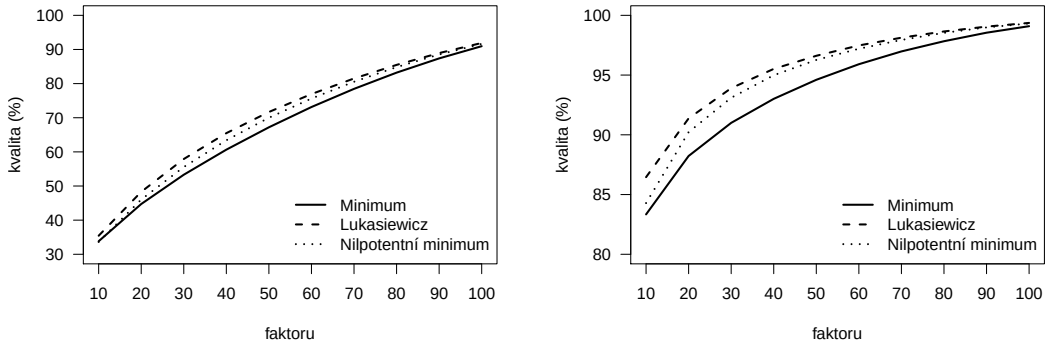
Tabulka 11. Průběh procenta pokrytých pozic z pohledu velikosti rozkladů. Pro jednotlivé struktury a počty faktorů uvádí průměrné procento pokrytých pozic rozkladů s daným počtem faktorů.

struktura	10 f	20 f	30 f	40 f	50 f	60 f	70 f	80 f	90 f	100 f
minimum	34	45	53	61	67	73	78	83	87	91
step3	34	45	53	61	67	73	78	83	87	91
step5	34	45	53	61	67	73	78	83	87	91
step15	34	45	53	61	67	73	78	83	87	91
step17	34	45	53	61	67	73	79	83	87	91
step51	34	45	54	61	68	73	79	83	87	91
step85	35	46	55	62	68	74	79	84	88	91
Łukas.	35	48	58	65	72	77	82	85	89	92
nilp. min.	34	46	56	63	70	76	81	85	89	92
bottomfew	34	45	54	61	68	73	79	83	87	91
densemid	34	45	54	61	68	73	79	83	87	91
sparsemid	34	45	54	61	67	73	79	83	87	91

Pro představu co naměřená data znamenají v praxi jsem vybral dva klasické obrázky 4. a 5. a jeden uměle vytvořený obrázek 6. a obrázky jsem zrekonstruoval

Tabulka 12. Průběh blízkosti z pohledu velikosti rozkladů. Pro jednotlivé struktury a počty faktorů uvádí průměrné procento blízkosti rozkladů s daným počtem faktorů.

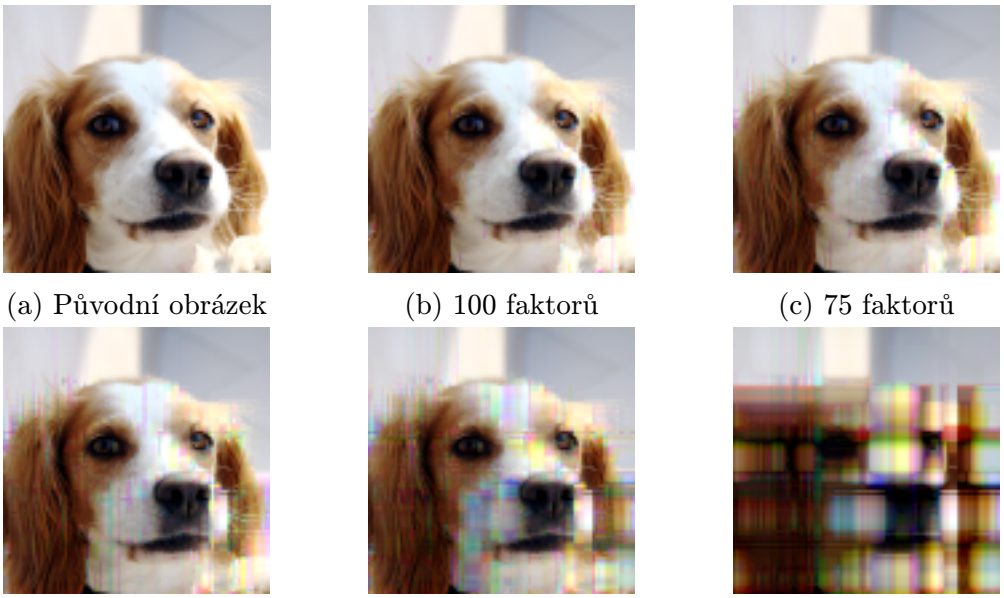
struktura	10 f	20 f	30 f	40 f	50 f	60 f	70 f	80 f	90 f	100 f
minimum	83	88	91	93	95	96	97	98	99	99
step3	83	88	91	93	95	96	97	98	99	99
step5	83	88	91	93	95	96	97	98	99	99
step15	83	88	91	93	95	96	97	98	99	99
step17	83	88	91	93	95	96	97	98	99	99
step51	83	88	91	93	95	96	97	98	99	99
step85	84	89	91	93	95	96	97	98	99	99
Łukas.	86	91	94	96	97	97	98	99	99	99
nilp. min.	84	90	93	95	96	97	98	99	99	99
bottomfew	84	89	91	93	95	96	97	98	99	99
densemid	83	88	91	93	95	96	97	98	98	99
sparsemid	83	88	91	93	95	96	97	98	98	99



(a) Pokryté pozice (b) Blízkost

Obrázek 3. Průběh rozkladu z pohledu počtu faktorů

z rozkladu pro Łukasiewiczovu strukturu pro různý počet faktorů. Úplný rozklad obrázku 4. měl v jednotlivých kanálech počty faktorů 204, 198 a 205. Osobně bych na první pohled označil za kvalitní až verzi pro 100 faktorů. Bohužel i zde jsou při bližším prozkoumání patrné artefakty v podobě fialových bodů a čar. Při této velikosti je z pohledu faktorové analýzy již každý atribut nahrazen faktorem. Na obrázku 5., jehož úplný rozklad měl v jednotlivých kanálech počty faktorů 177, 173 a 185, jsou artefakty ještě znatelnější. Úplný rozklad obrázku 6. měl 135 faktorů. Zde bych ani verzi pro 100 faktorů neoznačil za kvalitní, ale algoritmus zde měl těžkou úlohu, jelikož geometrie obrázku je pozorovateli jasná a i drobné chyby jsou snadno viditelné.

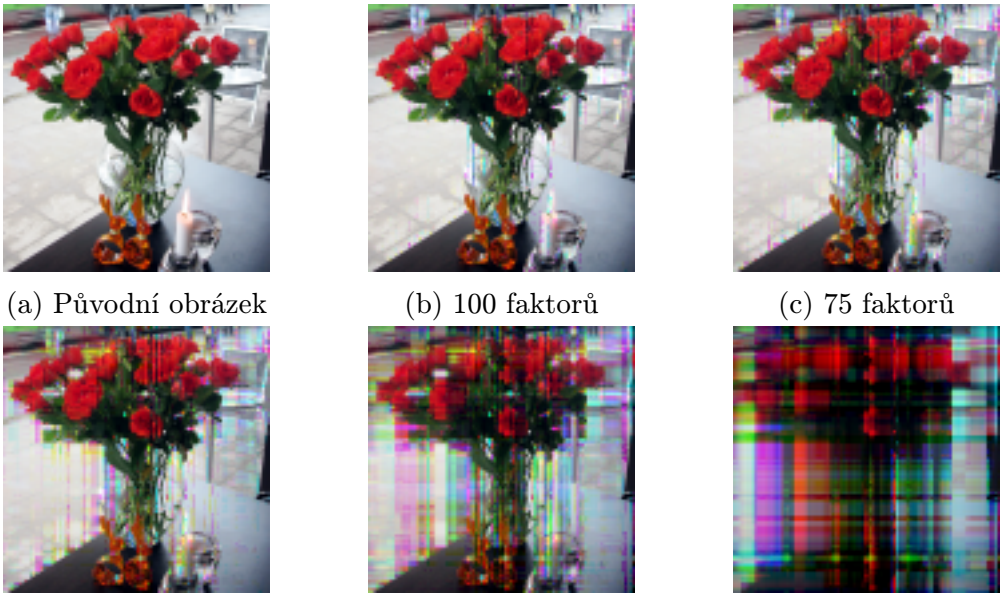


(a) Původní obrázek (b) 100 faktorů (c) 75 faktorů

(d) 50 faktorů (e) 25 faktorů (f) 5 faktorů

Obrázek 4. Rekonstrukce klasického obrázku pro různý počet faktorů

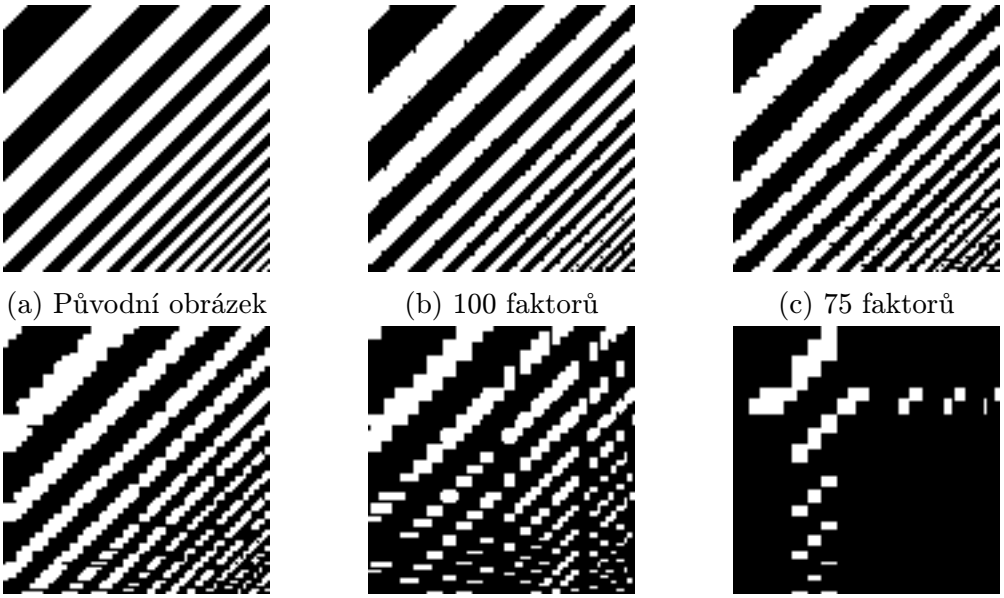
Pro vizuální představu jak vypadají samotné faktory a jak probíhá jejich skládání uvádím obrázek 7. Na obrázku jsou vykresleny první tři faktory, tak jak je algoritmus vybral při vytváření rozkladu pro Łukasiewiczovu strukturu. Faktory pro takto složitý obrázek vypadají velmi abstraktně a lze je jen stěží vizuálně interpretovat. Dále je na obrázku vykreslen rekonstruovaný obrázek po postupném přidávání těchto faktorů do rozkladu (po přidání prvního faktoru je samozřejmě rekonstruovaný obrázek roven tomuto faktoru). Každý faktor tvoří podobrázek rozloženého obrázku a jejich kombinací se k němu postupně blížíme. Tato kombinace faktorů je realizována operací maximum po jednotlivých složkách. Stupeň 0 odpovídá černé barvě a stupeň 255 odpovídá bílé barvě. Kombinací faktorů tedy dochází k překrývání světlejšími tóny, což je dobře patrné z obrázku 6. Po třech faktorech se již na obrázku 7. začíná rýsovat původní obrázek.



(a) Původní obrázek (b) 100 faktorů (c) 75 faktorů

(d) 50 faktorů (e) 25 faktorů (f) 5 faktorů

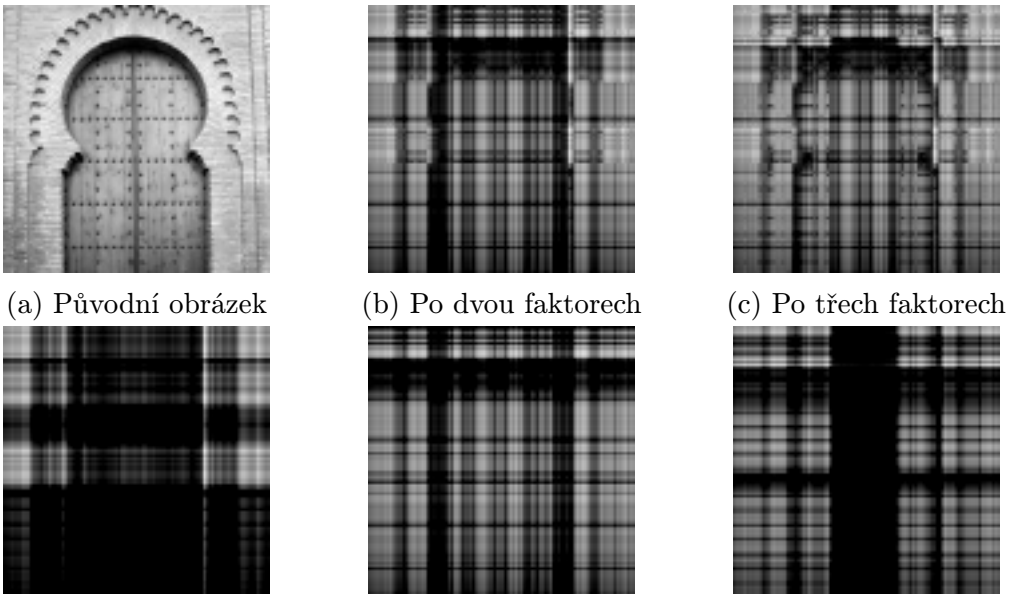
Obrázek 5. Rekonstrukce klasického obrázku pro různý počet faktorů



(a) Původní obrázek (b) 100 faktorů (c) 75 faktorů

(d) 50 faktorů (e) 25 faktorů (f) 5 faktorů

Obrázek 6. Rekonstrukce uměle vytvořeného obrázku pro různý počet faktorů



(a) Původní obrázek (b) Po dvou faktorech (c) Po třech faktorech

(d) První faktor (e) Druhý faktor (f) Třetí faktor

Obrázek 7. Postupné překrývání faktorů

5.3. Vhodné a nevhodné typy obrázků z hlediska komprese

Obecně se metodě daří na obdélníkovitých tvarech, které jsou vodorovné s okraji obrázku, což je patrné z vizuální reprezentace faktorů na obrázku 7. Na obrázku 8. jsou uvedeny obrázky velikosti 100×100, které měly nejmenší úplné rozklady. Velikosti úplných rozkladů pro Łukasiewiczovu strukturu jsou uvedeny v tabulce 13.a. Ostatní struktury vedly na podobné velikosti rozkladů. Průměrná velikost rozkladů v Łukasiewiczově struktuře pro obrázky velikosti 100×100 byla 158, jak bylo uvedeno v tabulce 9.

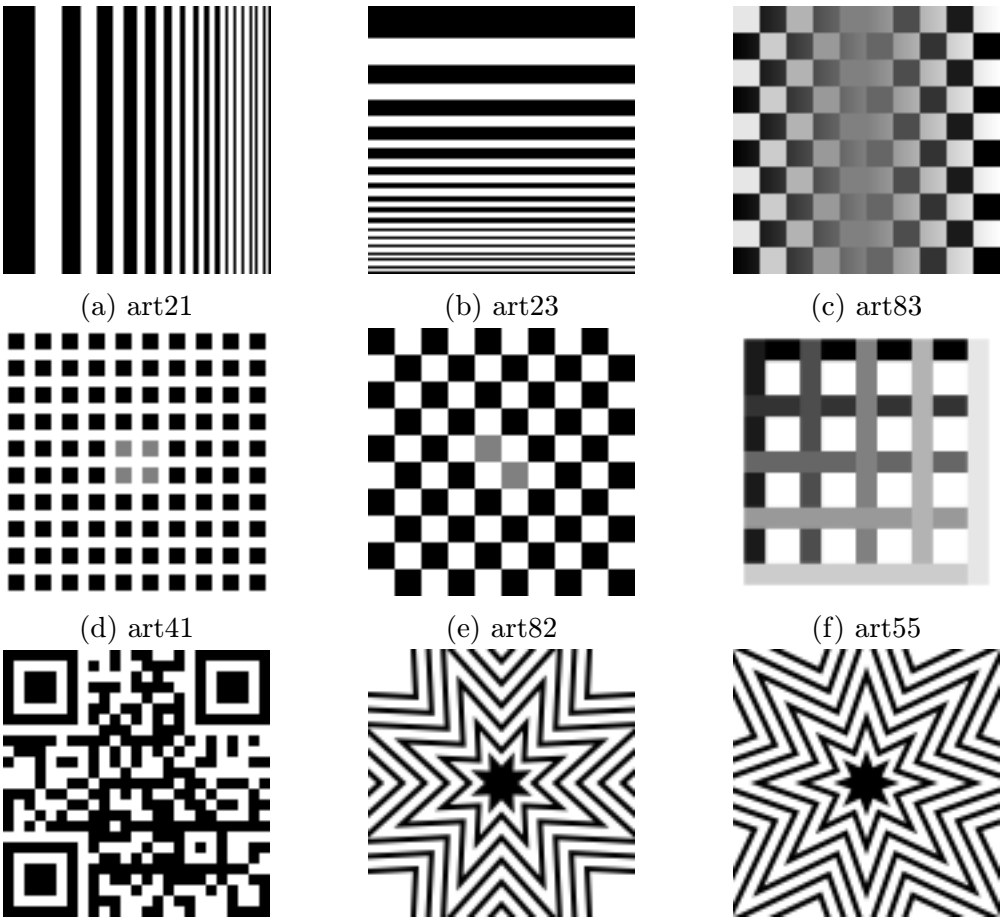
Tabulka 13. Počty faktorů úplných rozkladů vhodných a nevhodných obrázků.

obrázek	faktorů
art21	1
art23	1
art83	2
art41	3
art82	3
art55	14
QR kód	32
art35	76
art33	86

(a) Vhodné obrázky

obrázek	faktorů
Hrad	190 205 205
Pes	204 198 205
Křídlo	195 203 214
art57	196
art22	135
art73	171

(b) Nevhodné obrázky



(a) art21 (b) art23 (c) art83

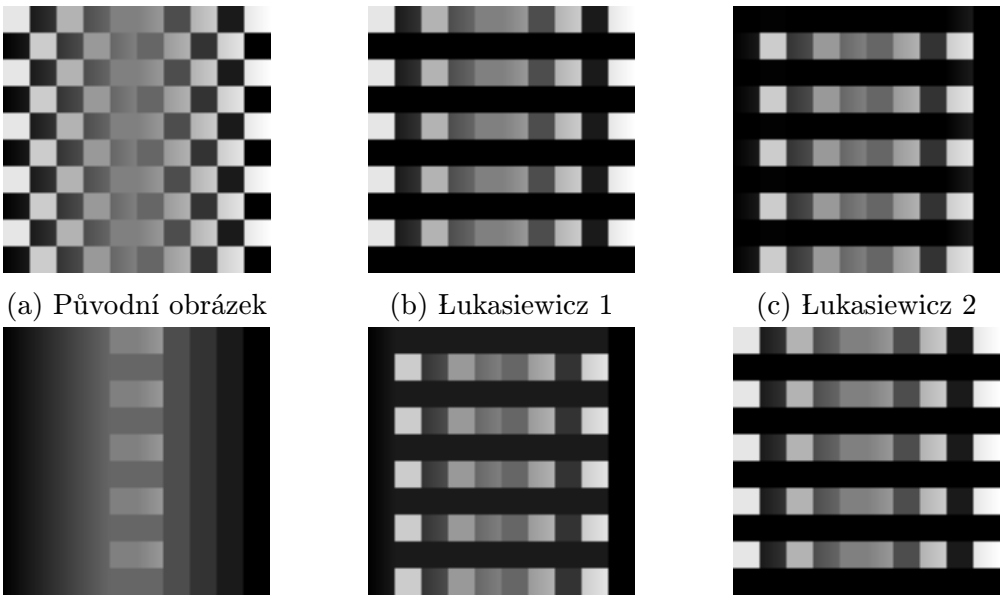
(d) art41 (e) art82 (f) art55

(g) QR kód (h) art35 (i) art33

Obrázek 8. Příklady vhodných obrázků

Obrázky 8.a a 8.b byly ve všech strukturách „rozloženy“ na jediný faktor. V těchto obrázcích se stále opakuje jeden řádek (případ obrázku 8.a) nebo sloupec (případ obrázku 8.b). Pro případ opakujícího se řádku stačí tento řádek vzít jako intent a odpovídající extent nastavit na samé jednotky, což řádek „nakopíruje“ po celé výšce obrázku. Obdobně pro případ opakujícího se sloupce, který vezmeme jako extent a intent bude obsahovat samé jednotky.

Obrázek 8.c již vyžaduje alespoň dva faktory. Pokud obrázek rozdělíme na sudé a liché vodorovné pruhy, pak jde opět o podobrázky, ve kterých se opakuje jeden řádek, a lze tedy každý z nich reprezentovat jedním faktorem. Šedý efekt, který na obrázku způsobuje postupné „šednutí“ směrem ke středu obrázku, principiálně obrázek nekomplikuje, jelikož stačí snížit nebo zvýšit patřičné stupně v intentu, ale může pro některé struktury algoritmus zmást a donutit ho vybrat nevhodný faktor, který aktuálně pokrývá nejvíc pozic, ale později bude stejně celý nahrazen kombinací jiných faktorů, které budou muset být do rozkladu stejně přidány. Algoritmus tomuto „odolal“ pro Łukasiewiczovu strukturu a nilpotentní minimum, pro které měly úplné rozklady dva faktory. Ostatní struktury vedly na tři faktory, kde právě první vybraný faktor reprezentoval převážně onen šedý efekt a další dva faktory stejně reprezentovaly sudé a liché pruhy. Příklad faktorů pro Łukasiewiczovu strukturu a minimum je uveden na obrázku 9. Toto je cena, kterou algoritmus platí za to, že koncepty vybírá postupně po jednom místo hledání nejmenší podmnožiny konceptů (podmnožin je exponenciálně mnoho k počtu konceptů).



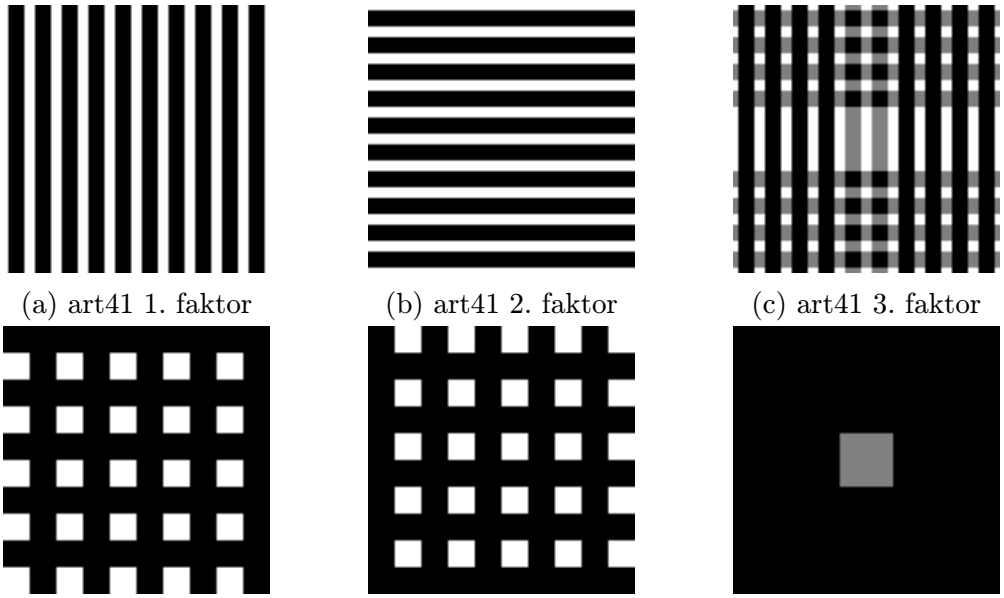
(a) Původní obrázek (b) Łukasiewicz 1 (c) Łukasiewicz 2

(d) Minimum 1 (e) Minimum 2 (f) Minimum 3

Obrázek 9. Faktory struktur Łukasiewicz a minimum pro obrázek art83

Obrázky 8.d a 8.e byly ve všech strukturách rozloženy na tři faktory. Faktory

těchto dvou obrázků pro Łukasiewiczovu strukturu jsou uvedeny na obrázku 10. Mohlo by se zdát, že obrázek 8.d by šel rozložit na faktor reprezentující šedý střed a faktor reprezentující černé čtverečky. Musíme si ale uvědomit, že bílá barva je stupeň 255. Algoritmus tedy začíná na černém obrázku a musí do něj přidat bílou mřížku, kterou už nelze reprezentovat jedním konceptem. Intent by musel obsahovat samé jednotky, aby reprezentoval bílé pruhy, což mu znemožňuje alternovat bílou a černou v ostatních pruzích. Kdyby byly barvy prohozeny a obrázek tedy obsahoval bílé čtverečky na černém pozadí, pak by již stačily dva koncepty, jeden na šedý střed a druhý by v intentu alternoval bílou a černou barvu pro čtverečky a v extentu by alternoval bílou a černou barvu pro zachování pruhů se čtverečky a vynulování černých pruhů.



(a) art41 1. faktor (b) art41 2. faktor (c) art41 3. faktor

(d) art82 1. faktor (e) art82 2. faktor (f) art82 3. faktor

Obrázek 10. Faktory obrázků art41 a art82

Na obrázcích 8.h a 8.i jsem chtěl demonstrovat, že velké obdélníky pomáhají jen když jsou vodorovné s okraji obrázku. Obrázek 8.i je pouhým pootočením obrázku 8.h, což narušilo vodorovnost mnoha bílých obdélníků a počet faktorů vzrostl ze 76 na 86. Extrémním příkladem jsou pak obrázky 8.a a 11.e, kde počet faktorů vzrostl z 1 na 135.

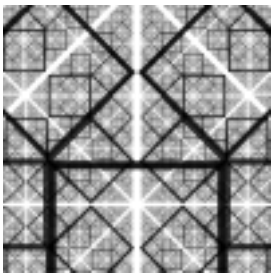
Na obrázku 11. uvádím příklady obrázků co dopadly hůře. Počty faktorů jejich úplných rozkladů pro Łukasiewiczovu strukturu jsou uvedeny v tabulce 13.b. První tři obrázky jsou klasické obrázky, které dopadly o něco hůř než ostatní. Nevidím v nich žádný společný faktor, který by rozklady vyloženě kazil. Ony se zas tak moc od průměrného počtu faktorů 158 neliší. Pokud vezmu v potaz jen klasické obrázky (fotky), pak se průměr pohybuje okolo 175 faktorů pro Łukasiewiczovu strukturu. U obrázku 11.d jde pak čistě o jeho složitost.

Obrázky 11.e a 11.f uvádím jako příklad obrázků, které by se mohly zdát jako jednoduché, ale přesto dopadly podobně jako klasické fotky. Pro srovnání ještě uvádím, že úplný rozklad v Łukasiewiczově struktuře pro náhodně vygenerovaný bílí šum vedl na 177 faktorů.

(a) Hrad

(b) Pes

(c) Křídlo



(d) art57

(e) art22

(f) art73

Obrázek 11. Příklady nevhodných obrázků

Metoda si tedy vede dobře pouze pro skutečně jednoduché obrázky. I mírně složitější obrázky typicky dopadnou podobně jako fotky. Pro klasické fotky a různou grafiku nelze očekávat příliš dobrou kompresi.

5.4. Snížení počtu stupňů

Algoritmus promising columns při hledání nejlepšího rozšíření aktuálního konceptu prochází mimo jiné všechny stupně a pro každý z nich vypočítá nový koncept, což je náročná operace. Provedl jsem tedy experiment se sníženým počtem stupňů. Testoval jsem na klasických obrázcích (fotkách) velikosti 100×100. Vynechal jsem černobílé obrázky, protože zde by pochopitelně nedošlo k žádné ztrátě kvality. K dispozici jsem měl 28 obrázků, z nichž polovina byla barevných, celkem jsem tedy získal data na 56 rozkladech.

Snížení počtu stupňů jsem provedl tak, že jsem si zvolil tzv. redukční faktor, což je číslo vyjadřující kolik po sobě jdoucích původních stupňů bude redukováno na nový jeden stupeň. Pro redukční faktor R jsem redukci původního obrázku a zpětné naškálování stupňů provedl jako

$$v'_{or} = \left\lfloor \frac{v_{or}}{R} \right\rfloor, \quad v_{rk} = v'_{rk} \cdot R + \frac{R}{2}, \quad v_{or}, v_{rk} \in L, \quad v'_{or}, v'_{rk} \in L',$$

kde L je množina původních stupňů, L' je nová množina s menším počtem stupňů, v_{or} je hodnota v původním obrázku, v'_{or} je odpovídající hodnota v redukovaném obrázku, v'_{rk} je odpovídající hodnota v obrázku zrekonstruovaném z rozkladu redukovaného obrázku a v_{rk} je odpovídající hodnota v obrázku vzniklém zpětným naškálováním zrekonstruovaného obrázku. Například pro redukční faktor 8 zredukuji původní stupeň 162 na 20 a místo množiny stupňů $\{0, \ldots, 255\}$ pracuji s množinou stupňů $\{0, \ldots, 31\}$. Zpětně stupeň 20 naškáluji do původní množiny stupňů jako 164. Ke ztrátě informace zde dochází při redukci stupňů, kdy celý interval původních stupňů délky R abstrahuji na jeden stupeň a zapomenu tak původní pozici stupně v tomto intervalu. Při zpětném škálování jej pouze umístím do středu původního intervalu, což je odhad, kterým se snažím minimalizovat chybu.

Za redukční faktory jsem zvolil všechny netriviální dělitele čísla 256, tedy čísla 2^1, 2^2, …, 2^7, tak aby byly původní stupně rovnoměrně rozděleny mezi nové stupně. Kvalitu jsem měřil pouze jako blízkost, jelikož zpětné škálování stupňů téměř nikdy nevede přesně na původní stupně, ale pro menší redukční faktory vede na velmi blízké stupně, a poměr přesně pokrytých pozic by tak kvalitu značně zkresloval. Testy jsem provedl pouze pro Łukasiewiczovu strukturu.

Naměřená data jsou uvedena v tabulce 14. Na každém řádku tabulky jsou data pro redukční faktor uvedený v prvním sloupci tabulky. Druhý sloupec udává počet nových stupňů po redukci. Třetí sloupec udává průměrný počet faktorů úplných rozkladů. Čtvrtý sloupec udává průměrnou blízkost v procentech obrázků získaných zpětným škálováním stupňů obrázků získaných z úplných rozkladů. Pro každý rozklad jsem spočítal poměr

$$\frac{\frac{b_R}{100}}{\frac{v_R}{v_1}},$$

kde b_R je blízkost v procentech obrázku získaného zpětným škálováním pro redukční faktor R a v_R je velikost úplného rozkladu redukovaného obrázku pro redukční faktor R. Jde tedy o poměr kvality a velikosti a vyšší hodnoty znamenají lepší poměry. V pátém sloupci je uveden průměr těchto poměrů přes všechny rozklady. Šestý sloupec tabulky má obdobný význam jako čtvrtý sloupec, pouze se zde vychází z částečných rozkladů o padesáti faktorech. Poslední sloupec udává průměrný čas (user time) výpočtu přes všechny obrázky ve vteřinách změřený na stroji Intel(R) Core(TM) i7-2600 @ 3.40 GHz.

Velikost úplných rozkladů se kupodivu z počátku dokonce mírně zvětšuje. K prvnímu významnějšímu poklesu velikosti rozkladů dochází až pro redukční faktor 32. Blízkost významněji klesá až od redukčního faktoru 16. Významně lepších poměrů mezi kvalitou a velikostí rozkladu je dosaženo jen pro velké redukční faktory, kde už je kvalita bohužel příliš nízká. Nezdá se, že by snížení počtu stupňů mělo větší vliv na částečné rozklady než na rozklady úplné (sle-

Tabulka 14. Vliv redukce stupňů na rozklady. Pro daný redukční faktor je uveden počet stupňů, průměrný počet faktorů úplných rozkladů, průměrná blízkost úplných rozkladů, průměrný poměr úplných rozkladů, průměrná blízkost částečných rozkladů s padesáti faktory a průměrný čas výpočtu rozkladů.

redukční faktor	$\|L\|$	faktorů	blízkost (%)	poměr	blízkost 50 f. (%)	čas (s)
(bez redukce) 1	256	176	100	1	96,6	930
2	128	179	99,8	0,98	96,8	484
4	64	181	99,6	0,97	96,9	244
8	32	182	99,2	0,97	96,8	119
16	16	176	98,4	1,02	96,4	54
32	8	158	96,7	1,16	95,2	21
64	4	129	93,4	1,47	92,5	6,4
128	2	79	87,5	2,78	87,1	1,1

doval jsem pouze částečné rozklady o velikosti 50 faktorů). Blízkost částečných rozkladů víceméně kopíruje blízkost úplných rozkladů. Dochází zde akorát k zajímavému jevu, kdy se blízkost částečných rozkladů chvíli nepatrně zvyšuje než začne očekávaně klesat. Snížení počtu stupňů má ovšem významný vliv na dobu výpočtu. Čas výpočtu klesá postupně o něco rychleji než počet stupňů. Pokud však zanedbáme vyloženě velké redukční faktory, které vedou na velmi nekvalitní výsledky, dá se říct, že čas klesá úměrně s počtem stupňů.

Na obrázku 12. je příklad jednoho obrázku pro různé redukční faktory. Jde o rekonstrukce z úplných rozkladů, aby šlo na obrázku snadno pozorovat pouze vliv snížení počtu stupňů. Při snižování počtu stupňů dochází k tzv. color bandingu, který se projevuje na plochách s velmi plynulým přechodem mezi stupni. Snižováním počtu stupňů dochází ke stále větším rozdílům v barvě mezi sousedními pixely. Toto je obzvlášť patrné na obloze obrázku od redukčního faktoru 8 a dál, i když při bližším prozkoumání lze tento jev detekovat již pro redukční faktor 4. Za kvalitní bych stále považoval obrázek pro redukční faktor 4, při kterém je výpočetní čas asi čtvrtinový oproti případu bez redukce stupňů.

Obrázek 13. ukazuje, jak vypadají zpětně naškálované obrázky z částečných rozkladů s 50 faktory. Z tohoto obrázku lze vidět, že snížení počtu stupňů může mít i kladný vliv na kvalitu částečných rozkladů. Například zde se mi obrázek pro redukční faktor 8 jeví jako nejkvalitnější. Artefakty způsobené nepřesností rozkladu jsou v něm méně patrné než na obrázcích pro nižší redukční faktory. Z pohledu artefaktů se možná za ještě kvalitnější jeví obrázek pro redukční faktor 32, zde se ale na druhou stranu už hodně projevuje malý počet barev.

Snížením počtu stupňů tedy nelze dosáhnout menších nebo výrazně kvalitnějších rozkladů. Pokud jsou ovšem přijatelné nepřesné rekonstrukce (ani úplné rozklady nebudou rekonstruovány přesně kvůli ztrátě informace během snížení počtu stupňů), lze dosáhnout mnohem kratších výpočetních časů a pokud počítáme částečné rozklady, nemusí být snížení počtu stupňů vůbec na úkor kvality.

(a) Původní obrázek

(b) Redukční faktor 2

(c) Redukční faktor 4

(d) Redukční faktor 8

(e) Redukční faktor 16

(f) Redukční faktor 32

(g) Redukční faktor 64

(h) Redukční faktor 128

Obrázek 12. Rekonstrukce obrázků z úplných rozkladů pro různé redukční faktory

(a) Původní obrázek

(b) Bez redukce

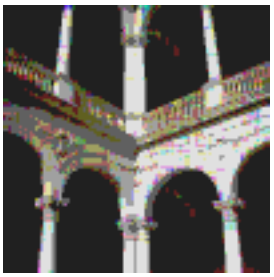
(c) Redukční faktor 2

(d) Redukční faktor 4

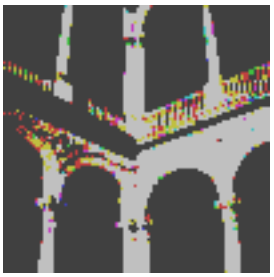
(e) Redukční faktor 8

(f) Redukční faktor 16

(g) Redukční faktor 32



(h) Redukční faktor 64



(i) Redukční faktor 128

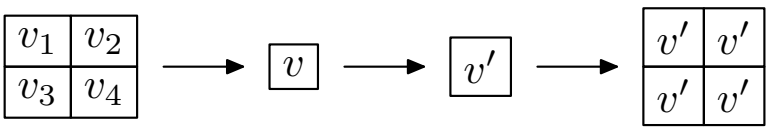
Obrázek 13. Rekonstrukce obrázků z částečných rozkladů s 50 faktory pro různé redukční faktory

Vhodnými redukčními faktory jsou faktory od 2 do 8, ale volba redukčního faktoru bude hodně záviset na daném obrázku.

5.5. Podvzorkování obrázků

Podobně jako se zmenšením počtu stupňů jsem provedl i experiment se zmenšením rozměru vstupního obrázku. Pro rozklad menších obrázků je typicky potřeba méně faktorů a faktory jsou zároveň menší. Například rozklad s 50 faktory obrázku velikosti 100×100 má absolutní velikost 10 kB. Do stejné absolutní velikosti rozkladu se vejde 100 faktorů pro obrázek velikosti 50×50. Pro částečné rozklady se tedy nabízí, zda není výhodnější vstupní obrázek podvzorkovat, pro tento menší obrázek spočítat kvalitnější rozklad a pak jej zpět převést na původní velikost. Při podvzorkování samozřejmě přijdeme o nějakou kvalitu, ale možná ji vynahradí kvalitnější částečný rozklad menšího obrázku.

Test jsem provedl na 28 fotkách velikosti 100×100 s celkem 56 kanály. Testoval jsem pouze Łukasiewiczovu strukturu. Podvzorkování jsem provedl tak, že jsem si zvolil redukční faktor R (konkrétně jsem zkusil redukční faktory 2 a 4) a každý čtverec původních hodnot o délce strany R jsem redukoval na jednu hodnotu rovnou průměru původních hodnot. Dostal jsem tak obrázky velikosti 50×50 a 25×25, na kterých jsem prováděl rozklady. Zpětné zvětšení na původní velikost jsem provedl tak, že z jedné hodnoty získané z rozkladu jsem vyrobil čtverec o délce strany R, ve kterém byly všechny hodnoty rovny hodnotě z rozkladu. Schéma tohoto procesu je znázorněno na obrázku 14. Hodnoty v_i jsou hodnoty v původním obrázku, v je vzorek vypočten jako $v = \frac{\sum_i v_i}{4}$ a v' je odpovídající hodnota reprezentovaná rozkladem.



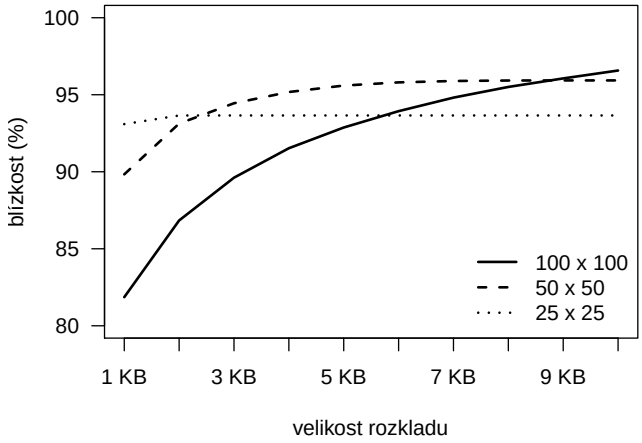
Obrázek 14. Schéma procesu podvzorkování

Naměřená data jsou uvedena v tabulce 15. Původní obrázky o velikosti 100×100 jsem podvzorkováním zmenšil a rozložil. Z rozkladu zrekonstruovaný obrázek jsem zvětšil na původní velikost a změřil blízkost k původnímu obrázku. Tabulka uvádí průměrné blízkosti přes všechny rozklady. Poměr pokrytých pozic by opět zkresloval kvalitu, proto jej neuvádím. Vždy jsem porovnával rozklady stejné absolutní velikosti. Například pro sloupec „5 kB“ jsem bral pro původní obrázek velikosti 100×100 rozklad s 25 faktory, pro obrázek velikosti 50×50 rozklad s 50 faktory a pro obrázek velikosti 25×25 rozklad se 100 faktory. Na obrázku 15. jsou hodnoty z tabulky proložené křivkou.

V nižších velikostech rozkladů podvzorkování skutečně pomáhá. Například rozklad s absolutní velikostí 1 kB pro obrázek velikosti 25×25 je z hlediska blíz-

Tabulka 15. Vliv podvzorkování obrázku na rozklady. Tabulka pro daný rozměr obrázku a danou velikost rozkladu v kilobajtech uvádí průměrnou blízkost z rozkladu zrekonstruovaných a na původní velikost zvětšených obrázků k původním obrázkům.

rozměr ob.	1 kB	2 kB	3 kB	4 kB	5 kB	6 kB	7 kB	8 kB	9 kB	10 kB
100×100	81,9	86,8	89,6	91,5	92,9	93,9	94,8	95,5	96,1	96,6
50×50	89,8	93,1	94,4	95,2	95,6	95,8	95,9	95,9	95,9	95,9
25×25	93,1	93,7	93,7	93,7	93,7	93,7	93,7	93,7	93,7	93,7



Obrázek 15. Srovnání blízkosti obrázků s různým stupněm podvzorkování

kosti průměrně zhruba stejně kvalitní jako pro obrázek velikosti 100×100 rozklad o velikosti 5 kB. Se zvyšující se velikostí rozkladů se kvalita rozkladů podvzorkovaných obrázků postupně ustálí a dále se nezlepšuje. Průměrné počty faktorů úplných rozkladů byly 176 pro obrázky velikosti 100×100, 83 pro obrázky velikosti 50×50 a 38 pro obrázky velikosti 25×25. Rozklady obrázků velikosti 25×25 tedy dosahují své maximální kvality kolem velikosti rozkladu 2 kB a rozklady obrázků velikosti 50×50 kolem velikosti 8 kB. Podvzorkováním se tedy spíše omezuje maximální možná kvalita, ale dokud jí není dosaženo, je podvzorkování z hlediska blízkosti výhodné.

Na obrázku 16.b je výše uvedeným způsobem zpětně zvětšený obrázek, který vznikl podvzorkováním původního obrázku 16.a na velikost 25×25. Na obrázku jsou patrné „kostičky“ pixelů velikosti 4×4 stejné barvy. Ty vznikají, pokud se zvětšuje výše uvedeným jednoduchým způsobem, a kazí vizuální dojem z obrázku. Zkusil jsem místo toho obrázek zvětšit programem `convert` s parametrem `-resize` a výsledek je zobrazen na obrázku 16.c. Program `convert` pravděpodobně implicitně používá při zvětšování některou ze supersamplingových metod. Nevýhodou tohoto přístupu je naopak viditelné rozostření obrázku. Výsledek je ale vizuálně přijatelnější, i když numericky bude obrázek průměrně vzdálenější. Metodu zpětného zvětšování lze samozřejmě volit různě.

Na obrázku 17. jsou srovnány dva rozklady velikosti 5 kB. Při této veli-


(a) Původní obrázek


(b) jednoduché zvětšení


(c) program convert

Obrázek 16. Dvě různé metody zpětného zvětšení

kosti dochází k poloviční kompresi původního obrázku velikosti 100×100. Obrázek 17.b je rekonstrukcí částečného rozkladu s 25 faktory bez podvzorkování. Obrázek 17.c vznikl zvětšením pomocí programu `convert` obrázku zrekonstruovaného z rozkladu velikosti 50 faktorů původního obrázku podvzorkovaného na velikost 50×50. Na jednu stranu podvzorkování umožní při stejné velikosti tvořit rozklady s větším počtem faktorů, což vede na méně artefaktů v obrázku, na druhou stranu je ale zase obrázek při rekonstrukci velmi rozostřen.


(a) Původní obrázek


(b) bez podvzorkování


(c) podvzorkování na 50×50

Obrázek 17. Rekonstrukce obrázku z rozkladu velikosti 5 kB

6. Diskuze

Přístup k faktorové analýze pomocí nástrojů formální konceptuální analýzy a použití konceptů za faktory je poměrně nová metoda, poprvé publikována v roce 2009. Metoda je zajímavá z teoretického hlediska a pravděpodobně si najde mnoho aplikací, ale čistě z hlediska komprese obrazu se mi nejeví jako vhodná.

Současné úspěšné kompresní algoritmy vychází převážně z poznatků o lidském zraku a využívají těchto znalostí k uchování co největšího množství kritické informace a redukci méně významné informace. Například metoda JPEG převádí data z prostorové domény do frekvenční domény a k největší redukci kvality dochází ve vysokých frekvencích, které lidský zrak nedokáže tak dobře vnímat. To vede na dobré kompresní poměry, aniž by se přitom obrázek nějak viditelně zhoršil. Zde prezentovaná metoda vybírá faktory, které pokryjí co nejvíc hodnot původní matice, bez ohledu na jejich „důležitost“. Nevýhoda tohoto přístupu je patrná například z obrázků 4. a 5. I pro velké počty faktorů stále dochází k viditelným artefaktům v obrázku, které velmi kazí celkový dojem z obrázku. Z hlediska průměrné blízkosti na pixel se pár nepokrytých a velmi odlišných hodnot nejeví jako kritické, ale ve výsledném obrázku se pak například objeví fialová čára na světlém pozadí. Z tohoto důvodu si myslím, že metoda není vhodná, pokud jde o zachování vizuální kvality obrazu vnímané člověkem. Naopak by metoda mohla najít uplatnění v oblastech strojového zpracování obrazu, kde jde o kvalitu měřenou v nějaké numerické metrice. Pro tento případ lze i výběr samotných konceptů upravit tak, aby co nejvíc vyhovoval dané metrice.

Ve výsledcích jsem uvedl, že celková „blízkost“ je vhodnější hodnocení kvality rozkladu než poměr pokrytých pozic. V algoritmu se přitom koncepty vybírají právě podle pokrytých pozic, i když by se dalo měřit, jak moc daný koncept přiblíží rozklad vstupnímu obrázku. Musíme si ale uvědomit, že při rekonstrukci obrázku se koncepty kombinují po složkách operací maximum. Operace maximum vybírá jednu hodnotu. I když tedy budeme mít k dispozici koncept, který rozklad hodně přiblíží k obrázku, muže později přidáním dalších konceptů dojít k překrytí původních hodnot a „přínos“ tohoto konceptu je rázem snížen. Z tohoto důvodu je podle mě rozumné vybírat koncepty na základě pokrytých pozic, což je jediná jistá kvalita, kterou koncept do rozkladu přinese. I když bude později do rozkladu přidán koncept, který zduplikuje pokrytí některé pozice, nebyla alespoň tato pozice při jeho výběru do kvality započítána, což zvyšuje šanci, že pokryje více dosud nepokrytých pozic.

Dalším problémem metody je pak její samotný výkon. Pro vyšší kvality metoda velmi často ani nekomprimuje, spíše naopak. Co se času výpočtu týče, i když vezmeme algoritmus promising columns, úplný rozklad běžného obrázku velikosti 100×100 trvá asi 10 minut (čas je zhruba úměrný počtu faktorů, takže ani částečné rozklady nebudou nijak rychlé). Čas navíc rychle roste s velikostí vstupního obrázku. Pro obrázky velikosti 128×128 to už je asi 25 minut. Pro vstupy s reálnou velikostí je algoritmus nepoužitelný. Prostor konceptů je prostě příliš velký

(i promising columns prohledává nemalou část prostoru konceptů). Velmi by pomohlo, kdyby se přišlo na způsob, jak přímo spočítat jen malý počet konceptů s velkým přínosem ke kvalitě rozkladu a algoritmus by pak vybíral jen z těchto konceptů.

Při případném nasazení metody v kompresním algoritmu je vhodné ji doplnit nějakým předzpracováním dat a efektivním uložením výsledného rozkladu, podobně jak to dělají současné kompresní algoritmy. V úvahu připadá například snížení počtu stupňů a podvzorkování vstupních dat a volba efektivního kódování pro uložení rozkladu. Z tohoto důvodu jsem neprovedl žádné srovnání s existujícími metodami, jelikož zde prezentovaná metoda by byla spíše jádrem nějakého potenciálního kompresního algoritmu.

Oblastí dalšího zkoumání metody by mohla být volba jiných barevných modelů než jen RGB, jako třeba modelů HSL nebo HSV, popřípadě převod obrázkových dat do frekvenční domény.

Závěr

Využití formálních konceptů jako faktorů pro faktorizaci matic je novým přístupem k faktorové analýze. Jednou z možných aplikací této metody je využití částečných rozkladů, které pouze aproximují rozkládanou matici, pro kompresi obrazu.

V diplomové práci jsem se zabýval možnostmi této aplikace a provedl jsem řadu experimentů, pro jejichž účel jsem vytvořil program implementující dva algoritmy provádějící faktorizaci matic. Program je katedře k dispozici pro další případný výzkum v této oblasti. Metoda se mi nejeví jako vhodná pro kompresi obrazu. Poměr komprese a kvality není dobrý a obraz zrekonstruovaný z částečných rozkladů i pro velké počty faktorů často trpí artefakty, které vážně kazí dojem z obrazu vnímaný člověkem. Spíše bych možné využití metody viděl v oblasti strojového zpracování obrazu, kde je kvalita obrazu měřena různými numerickými metrikami. Dalším nedostatkem metody je velikost konceptuálních svazů vstupních obrázků. Ukázalo se, že i obrázek velikosti 5×5 může mít reálně milióny konceptů a i efektivnější ze dvou známých algoritmů stále prochází nemalé množství těchto konceptů. To vede na příliš dlouhý čas výpočtu algoritmu, který znemožňuje případnou aplikaci metody v kompresním algoritmu.

Experimenty jsem provedl s množinou stupňů o velikosti 256. S takto velkou množinou stupňů nebyly dosud provedeny žádné experimenty. Z testů vyplynuly některé poznatky, jako třeba vliv volby struktury na oba algoritmy z hlediska velikosti a kvality rozkladů a doby výpočtu algoritmu, velikost konceptuálních svazů pro různé struktury, vhodné a nevhodné typy obrázků pro rozklad, jak rychle se zvyšuje kvalita přidáváním dalších faktorů, vliv snížení počtu stupňů a vliv podvzorkování. Některé z těchto poznatků byly již známé.

Nestihl jsem provést některé další experimenty jako třeba podívat se na vliv snížení počtu stupňů a podvzorkování pro další struktury než jen Łukasiewiczovu, volbu jiných barevných modelů než jen modelu RGB nebo převod obrázkových dat do frekvenční domény. I přesto jsem dosáhl všech cílů práce.

Conclusions

The use of formal concepts as factors for matrix decomposition is novel approach to factor analysis. One possible application of this method is utilization of partial decompositions which merely approximate decomposed matrix for the purpose of image compression.

In this diploma thesis I have studied the potential of this application and have done numerous experiments. I have developed a program implementing two algorithms for matrix decomposition for the purpose of experiments. This program is available to department for potential further research in this area. I do not see the method as being suitable for image compression. Its compression to quality ratio is not good and image reconstructed from partial decompositions often suffers from artifacts even for high number of factors, which severely corrupts human perception of the image. The method is in my opinion more suitable in areas of machine image processing, where image quality is measured by various numeric metrics. Further drawback of the method is the size of the conceptual lattices for input images. It turned out, that even image of size 5×5 can realistically have millions of concepts and even the more efficient of the two known algorithms still has to process a large number of those concepts. That leads to far too long computation time, which disables potential use of the method in compression algorithm.

I have done the experiments with set of grades of size 256. None experiments have been done with such large set of grades yet. Couple observations have emerged from these experiments, such as structure choice impact on both algorithms influencing size and quality of decompositions and computation time, size of the conceptual lattices for different structures, types of images suitable and not suitable for decomposition, how fast does the quality increase by adding further factors, impact of reducing number of grades and impact of subsampling. Some of these observations were already known.

I have not managed to do in time some further experiments such as observing impact of reducing number of grades and subsampling for other structures than just Łukasiewicz one, choice of other color models than just RGB model or transforming image data into frequency domain. I have still achieved all goals of this thesis.

Reference

[1] Asuni, N.: Sada testovacích obrázků. http://testimages.sf.net, Duben 2011, [Online], [cit. 2013-04-13].

[2] Bartholomew, D. J.; Knott, M.: *Latent Variable Models and Factor Analysis*. London: Arnold, druhé vydání, 1999, 224 s.

[3] Bělohlávek, R.; Vychodil, V.: *Fuzzy Equational Logic*, *Studies in Fuzziness and Soft Computing*, ročník 186. Springer, 2005, ISBN 978-3-540-26254-1, 1-266 s.

[4] Bělohlávek, R.; Vychodil, V.: Factor Analysis of Incidence Data via Novel Decomposition of Matrices. In *ICFCA*, *Lecture Notes in Computer Science*, ročník 5548, editace S. Ferré; S. Rudolph, Springer, 2009, ISBN 978-3-642-01814-5, s. 83–97.

[5] Bělohlávek, R.; Vychodil, V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. Syst. Sci.*, ročník 76, č. 1, 2010: s. 3–20.

[6] Eastman Kodak Company: Sada testovacích obrázků od společnosti Kodak. http://r0k.us/graphics/kodak/, 1999, [Online], [cit. 2013-04-13].

[7] Ganter, B.; Wille, R.: *Formal concept analysis - mathematical foundations*. Springer, 1999, ISBN 978-3-540-62771-5, I-X, 1-284 s.

[8] Golub, G. H.; van Loan, C. F.: *Matrix computations*. Johns Hopkins University Press, třetí vydání, 1996, ISBN 978-0-8018-5414-9, I-XXVII, 1-694 s.

[9] Klement, E.-P.; Mesiar, R.; Pap, E.: *Triangular Norms*, *Trends in Logic*, ročník 8. Springer, 2000, ISBN 978-0-7923-6416-0, 387 s.

[10] Lee, D. D.; Seung, H. S.: Learning the parts of objects by non-negative matrix factorization. *Nature*, ročník 401, 1999: s. 788–791.

[11] McDonald, R. P.: *Factor Analysis and Related Methods*. Lawrence Erlbaum Associates, 1985, 259 s.

[12] Ming Hsieh Department of Electrical Engineering, USC Viterbi School of Engineering: Databáze obrázků USC-SIPI. http://sipi.usc.edu/database/, 1977, [Online], [cit. 2013-04-13].

[13] Outrata, J.; Vychodil, V.: Fast algorithm for computing fixpoints of Galois connections induced by object-attribute relational data. *Inf. Sci.*, ročník 185, č. 1, 2012: s. 114–127.

A. Dokumentace programu gfd

A.1. Použití

```
gfd [volby] INFILE OUTFILE
```

A.2. Popis

Program `gfd` rozkládá vstupní obrázek na faktory s využitím formálních konceptů za účelem (potenciálně ztrátové) komprese. Program načte obrázek `INFILE` a zapíše jeho rozklad do `OUTFILE` v interním formátu programu `gfd`. V dekompresním módu (parametr `-d`) načte `gfd` rozklad `INFILE` a vypočítá z něj odpovídající obrázek, který uloží do `OUTFILE` (formát obrázku je dán příponou).

A.3. Parametry

Parametry `-q`, `-f`, `-a` a `-i` ovlivňují kompresní mód (`-d` neuvedeno). Podobně, parametry `-s` a `-t` ovlivňují dekompresní mód (`-d` uvedeno). Parametr `-r` ovlivňuje oba módy.

-r Vypíše počty faktorů v rozkladu pro každý kanál obrázku. Při použití Set Cover algoritmu v kompresním módu se ještě před počty faktorů vypíše pro každý kanál řádek se třemi čísly. První číslo je počet konceptů v konceptuálním svazu. Druhé číslo je počet kandidátních konceptů (koncepty s unikátní maticí pokrytí). Poslední číslo je počet kolizí při hashování kandidátních konceptů (určeno pro sledování výkonu).

-d Přepíná do dekompresního módu. V argumentu `INFILE` je očekáván rozklad vytvořený programem `gfd`.

A.3.1. Parametry ovlivňující kompresní mód

-q QUALITY Minimální požadovaná kvalita rozkladu v procentech. Nižší hodnoty vedou na lepší kompresi. Možné hodnoty pro argument `QUALITY` jsou celá čísla od 1 do 100. Implicitní hodnota je 100.

-f FACTORS Maximální počet faktorů v rozkladu pro každý kanál obrázku. Nižší hodnoty vedou na menší rozklady na úkor kvality. Možné hodnoty argumentu `FACTORS` jsou kladná celá čísla. Implicitně není počet faktorů limitován.

Pokud jsou uvedeny oba parametry `-q` a `-f`, hledání rozkladu je ukončeno jakmile je splněna libovolná z těchto dvou podmínek.

-a ALGORITHM Zvolí použitý algoritmus. Možné hodnoty pro argument `ALGORITHM` jsou `sc` pro algoritmus set cover nebo `pc` pro algoritmus promising columns. Implicitně je použit algoritmus promising columns.

-i STRUCTURE Struktura reziduovaného svazu definující operace násobení $\otimes$ a reziduum $\rightarrow$. Možné hodnoty pro argument `STRUCTURE` jsou buď `luk` (Łukasiewicz), `god` nebo `min` (Gödel, taky zvaná jako minimum) a `nilmin` (Nilpotentní minimum) pro pojmenované struktury, nebo posloupnost čárkou oddělených idempotentů (např. `0,50,100,255`) udávající ordinální součet. Idempotenti musí být celá čísla od 0 do 255. Posloupnost musí obsahovat alespoň stupeň 0 a největší stupeň (jednotku operace $\otimes$, typicky 255). Největší idempotent je považován za největší stupeň. V případě pojmenovaných struktur (`luk` atd.) je největším stupněm 255. Implicitně je zvolena Łukasiewiczova struktura pro algoritmus promising columns, nebo minimum pro set cover.

Posloupnost stejně od sebe vzdálených idempotentů lze vygenerovat příkazem `seq -s , OD KROK DO`. Například posloupnost `0,100,200,255` lze zadat jako `-i` `` `seq -s , 0 100 255` ``,255.

A.3.2. Parametry ovlivňující dekompresní mód

-sN Navíc vygeneruje posloupnost obrázků reprezentující postupné přidávání faktorů. Jeden obrázek je vytvořen pro každý faktor. Jeho jméno je složeno z argumentu `OUTFILE` a pořadového čísla, ve kterém byl faktor vybrán během hledání rozkladu. Tento obrázek reprezentuje, co daný faktor do obrázku přidává. Další obrázky jsou vytvořeny pro celkový obraz po přidání každého faktoru. Jejich jména jsou vytvořena obdobným způsobem jako u obrázků faktorů, navíc s předponou „upto“ před pořadovým číslem. Tento obrázek reprezentuje celkový obraz po přidání prvních `K` faktorů, kde `K` je pořadové číslo uvedeno v názvu tohoto obrázku. Argument `N` je volitelný (použití např. `-s10` nebo jen `-s`) a pokud je uveden, posloupnost obrázků je generována jen pro prvních `N` faktorů. Uvedení 0 za `N` má stejný význam jako neuvedení parametru `-s`, tedy sekvence obrázků nebude generována. Tento parametr může snadno způsobit vygenerování stovek obrázků, takže je třeba jej používat opatrně.

-tPATH Navíc uloží textovou reprezentaci rozkladu. Argument `PATH` je volitelný (použití např. `-tmyfile.txt` nebo jen `-t`) a udává kam se má textová reprezentace uložit. Pokud není argument `PATH` uveden, místo vytvoření jakýchkoliv obrázků pouze uloží textovou reprezentaci do `OUTFILE`. Rozklad je zapsán jako posloupnost číselných matic. První je uvedena matice vypočítaného obrázku, potom matice extentů a poslední matice intentů (matice

jsou odděleny prázdným řádkem). Toto je provedeno pro každý kanál. Například textová reprezentace rozkladu RGB obrázku bude obsahovat 9 matic, první tři pro červený kanál, další tři pro zelený kanál a nakonec poslední tři pro modrý kanál.

A.4. Příklady použití

Minimální použití se všemi implicitními hodnotami je

```
gfd img.png dcmp.gfd
```

pro vytvoření rozkladu `dcmp.gfd` obrázku `img.png` a

```
gfd -d dcmp.gfd rimg.png
```

pro zpětnou rekonstrukci rozkladu `dcmp.gfd` na obrázek `rimg.png`. Pro nejmenší částečný rozklad s kvalitou alespoň 50 % v Gödelově struktuře a s výpisem počtu faktorů použijeme například

```
gfd -q 50 -i god -r img.jpg dcmp.gfd
```

Pro rekonstrukci ve formátu `tiff` a s vygenerováním prvních deseti faktorů použijeme například

```
gfd -ds10 dcmp.gfd rimg.tiff
```

Pro volbu algoritmu set cover, výpis sledovaných statistik a velikost rozkladu maximálně 50 faktorů v Gödelově struktuře nad množinou stupňů $\{0, 1, 2, 3, 4\}$ (obrázek by měl obsahovat pouze hodnoty do stupně 4, aby měl rozklad smysl) použijeme například

```
gfd -ra sc -f 50 -i 0,1,2,3,4 prettydark.png dcmp.gfd
```

Pro rekonstrukci s výpisem počtu faktorů, vygenerováním všech faktorů a uložením textové reprezentace do souboru `dcmp.txt` použijeme například

```
gfd -drs -tdcmp.txt dcmp.gfd rimg.png
```

Pro pouhé vypsání rozkladu do souboru `dcmp.txt` bez vytvoření rekonstruovaného obrázku použijeme například

```
gfd -dt dcmp.gfd dcmp.txt
```

Pro nejmenší částečný rozklad s kvalitou alespoň 85 %, ale maximálně se 100 faktory (i za cenu nesplnění požadované kvality), ve struktuře s operací $\otimes$, která má idempotenty $\{0, 3, 6, \ldots, 255\}$ použijeme například

```
gfd -q 85 -f 100 -i `seq -s , 0 3 255` img.png dcmp.gfd
```

Pro vygenerování posloupnosti idempotentů jsme zde použili externí program `seq` dostupný na Unixových systémech.

B. Obsah přiloženého CD

`bin/`
: Linuxová binárka programu `gfd`.

`data/`
: Ukázková data použitá pro potřeby obhajoby práce.

`doc/`
: Text diplomové práce ve formátu PDF a ZIP archiv se zdrojovým textem tohoto dokumentu a všemi potřebnými zdroji.

`src/`
: Kompletní zdrojové texty programu.

`readme.txt`
: Instrukce pro instalaci a spuštění programu a požadavky pro její provoz.