Steganografie v paletových obrázcích

Andrew Kozlik

KA MFF UK

Reprezentace barev

- Barvy reprezentujeme jako uspořádané trojice $(r, g, b) \in \{0, 1, \ldots, 255\}^3$.
- Množinu všech barev lze uspořádat lexikograficky jako množinu trojic.
- Podobnost barev můžeme měřit pomocí Eukleidovské normy $\|(r_1, g_1, b_1) - (r_2, g_2, b_2)\| = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$.
- Existují i jiné reprezentace barev, které mohou být vhodnější pro měření podobnosti, např. CIELAB.
- CIELAB rozkládá barvy na jasovou složku a dvě barvonosné složky.
- Transformace mezi RGB a CIELAB je nelineární.

Paletové obrázky

- Paletový obrázek sestává z palety barev a z matice indexů.
- Paleta barev je množina $C = \{ c_i \mid i \in I \}$ indexovaná čísly $I = \{0, 1, \ldots, |C| - 1\}$.
- Barva pixelu je určena indexem na odpovídající pozici v matici indexů.
- Každý paletový obrázek má mnoho různých reprezentací určených uspořádáním palety.

Steganografie v paletových obrázcích

1. Volbou uspořádání palety.

2. Vkládání do matice indexů.

   (a) Paletu setřídíme podle jasu. Potom LSB embedding.

   (b) Paletu setřídíme lépe. Potom LSB embedding.

   (c) Vkládání s optimálním přiřazením parity.

Steganografie volbou uspořádání palety

- Množinu všech permutací palety S_C můžeme uspořádat lexikograficky.
- Každé permutaci přiřadíme pořadové číslo od 0 do $|C|! - 1$.
- Každému pořadovému číslu přiřadíme nějakou zprávu.
- Paleta s 256 barvami dává nosič s kapacitou $\log_2 256! \approx 1684\,\mathrm{bitů} \approx 1{,}5$ SMS zprávy.

Steganografie volbou uspořádání palety

+ Nezanechává žádnou vizuální stopu v obrázku.
  (Pouze vybíráme jednu z mnoha ekvivalentních reprezentací obrázku.)

– Omezená kapacita.

– Chaoticky uspořádaná paleta budí podezření.
  (Většinou bývá setříděna podle jasu, odstínu, anebo podle četnosti barvy.)

– Otevření obrázku a opětovné uložení má zpravidla za následek zničení zprávy.

Vkládání do matice indexů

2. (a) EzStego

- Paletu setřídíme podle jasu a matici indexů přečísluje.
- Bity zprávy vkládáme do nejnižších bitů jednotlivých indexů v matici.
- Změna nejnižšího bitu v indexu příliš neovlivní jas pixelu.
- Může vést k zásadní změně odstínu pixelu.

Vkládání do matice indexů

2. (b) Vkládání s lépe setříděnou paletou.
   - Stejně jako v 2. (a), ale paletu uspořádáme, aby každé dvě po sobě následující barvy v paletě byly co možná nejpodobnější.
   - Například budeme minimalizovat součet
   $$\sum_{i=0}^{|C|-2} \|c_i - c_{i+1}\|$$
   - Problém hledání takovéhoto uspořádání je shodný s problémem obchodního cestujícího.
   - NP-úplný problém.

Vkládání do matice indexů

2. (c) Vkládání s optimálním přiřazením parity

- Každé barvě v paletě přiřadíme hodnotu 0 nebo 1 (*paritu*) tak, aby nejpodobnější barva v paletě měla opačnou hodnotu.
- Potřebujeme-li, aby barva některého pixelu měla opačnou paritu, stačí ji změnit na nejpodobnější barvu v paletě.
- Tímto minimalizujeme velikost změn vyvolaných vkládáním.

Vkládání s optimálním přiřazením parity

- Zobrazení, které každé barvě přiřazuje paritu, budeme značit p.
- Zobrazení, které každé barvě přiřazuje nejpodobnější barvu v paletě, budeme značit f

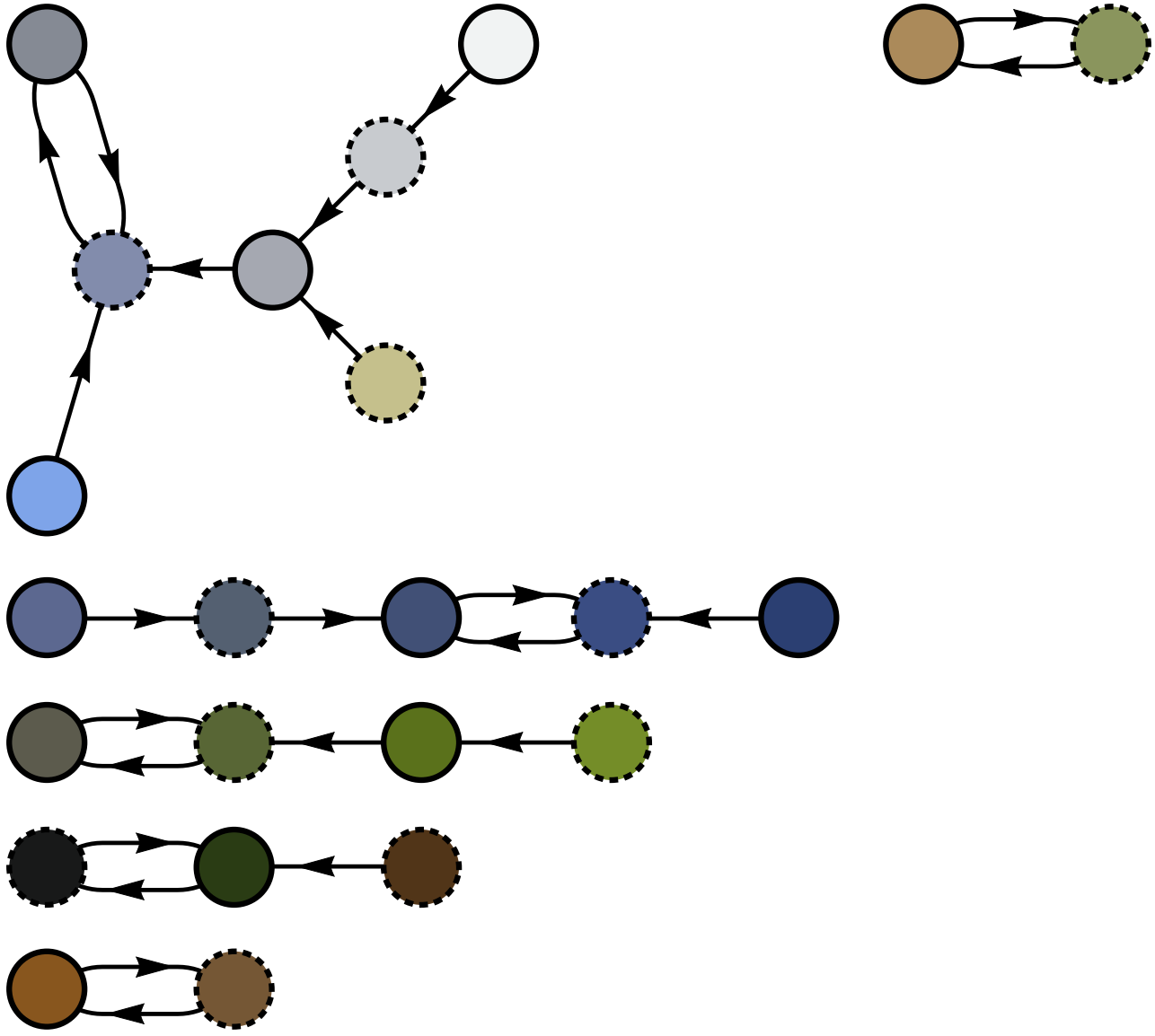
Definice

Nechť $C = \{ c_i \mid i \in I \}$ je paleta a (p, f) je dvojice zobrazení, kde $p : I \to \{0, 1\}$ a $f : I \to I$. Jestliže pro každé $i \in I$ platí

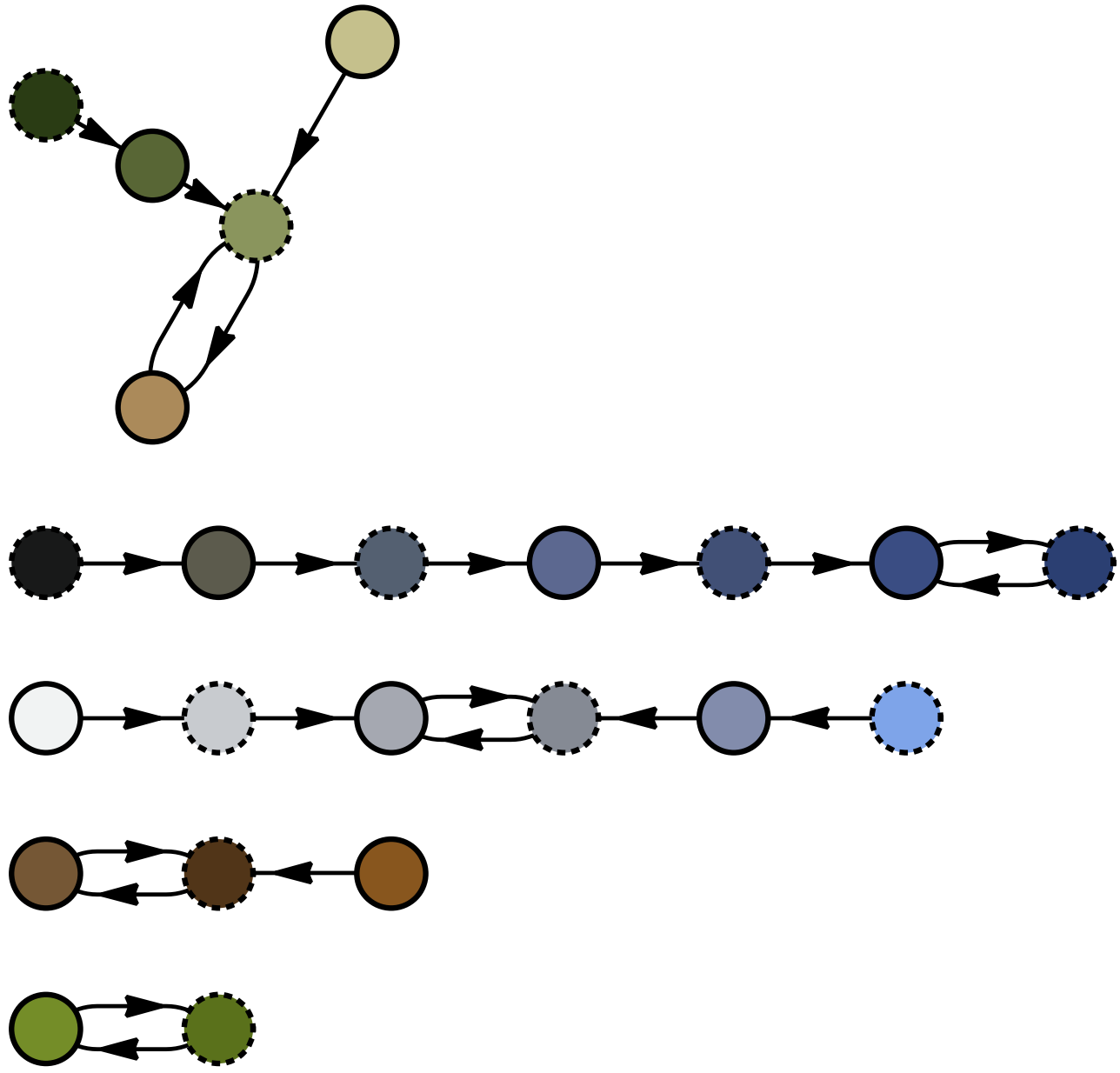
$$p(i) = 1 - p(f(i)) \quad \text{a} \quad \|c_i - c_{f(i)}\| = \min_{j \in I \setminus \{i\}} \|c_i - c_j\|,$$

pak říkáme, že (p, f) je *optimální přiřazení parity* pro paletu C.

Optimální přiřazení parity v RGB metrice



Optimální přiřazení parity v CIELAB metrice



Nejednoznačnost optimálního přiřazení parity

Optimální přiřazení parity existuje pro každou paletu, není však jednoznačně určeno.

- Nejednoznačnost p:
  Můžeme převrátit parity barev v libovolné komponentě a máme opět optimální přiřazení parity.
- Nejednoznačnost f:
  K jedné barvě můžou v paletě existovat dvě nejpodobnější barvy.
- Chceme jednoznačnost, protože (p, f) se v rámci komunikace neposílá. (Odesílatel i příjemce si ho spočítají samostatně ze znalosti palety.)
- Nejednoznačnosti se lze zbavit zavedením lineárního uspořádání na množině barev.

Lineární uspořádání barev

- Uspořádáme podle indexu v paletě?
- Problém uspořádání podle indexu:
  Zranitelné vůči přeuspořádání palety.
- Uspořádáme lexikograficky jako třísložkové vektory?
- Problém lexikografického uspořádání:
  Nelze rozlišit duplicitní barvy v paletě.
- Řešení: Uspořádáme lexikograficky a duplicitní barvy ztotožníme.

Algoritmus vkládání s optimálním přiřazením parity

vstup: nosič $x = (x_1, \ldots, x_n) \in I^n$
zpráva $z = (z_1, \ldots, z_m) \in \{0,1\}^m$
klíč $\pi \in S_n$
optimální přiřazení parity (p, f)
výstup: stegoobjekt $y = (y_1, \ldots, y_n) \in I^n$

1 **for** $i = 1, \ldots, n$ **do**
2 **if** $i > m$ **or** $p(x_{\pi(i)}) = z_i$ **then**
3 $y_{\pi(i)} := x_{\pi(i)}$
4 **else**
5 $y_{\pi(i)} := f(x_{\pi(i)})$
6 **return** y

Algoritmus extrakce s optimálním přiřazením parity

vstup: stegoobjekt $y = (y_1, \ldots, y_n) \in I^n$
klíč $\pi \in S_n$
délka zprávy m
optimální přiřazení parity (p, f)
výstup: zpráva $z = (z_1, \ldots, z_m) \in \{0,1\}^m$

1 **for** $i = 1, \ldots, m$ **do**
2 $\quad z_i := p(y_{\pi(i)})$
3 **return** z

Algoritmus optimálního přiřazení parity

vstup: paleta bez duplicit $C = \{ c_i \mid i \in I \}$
výstup: optimální přiřazení parity (p, f) pro C

1 $E := \{ (\|c_i - c_j\|, c_i, c_j) \mid i, j \in I, i \neq j \}$
2 **while** $E \neq \emptyset$ **do**
3 $(d, c_i, c_j) := \min_{\mathrm{LEX}} E$
4 **if** $p(j)$ is undefined **then**
5 $p(i) := 0$**;** $p(j) := 1$
6 $f(i) := j$**;** $f(j) := i$
7 $E := E \setminus (\mathbb{R} \times \{c_i, c_j\} \times C)$
8 **else**
9 $p(i) := 1 - p(j)$
10 $f(i) := j$
11 $E := E \setminus (\mathbb{R} \times \{c_i\} \times C)$
12 **return** (p, f)

Důkaz správnosti algoritmu

- Přiřazení $(d, c_i, c_j) := \min_{\mathrm{LEX}} E$ je dobře definované, jestliže paleta neobsahuje duplicity.

- Algoritmus nezávisí na uspořádání palety.
  - Nikde nedochází k porovnávání indexů i a j.

- Algoritmus skončí.
  - Množina E je konečná.
  - Při každém průchodu cyklem se zmenší alespoň o jeden prvek (d, c_i, c_j).

Důkaz správnosti algoritmu (pokračování)

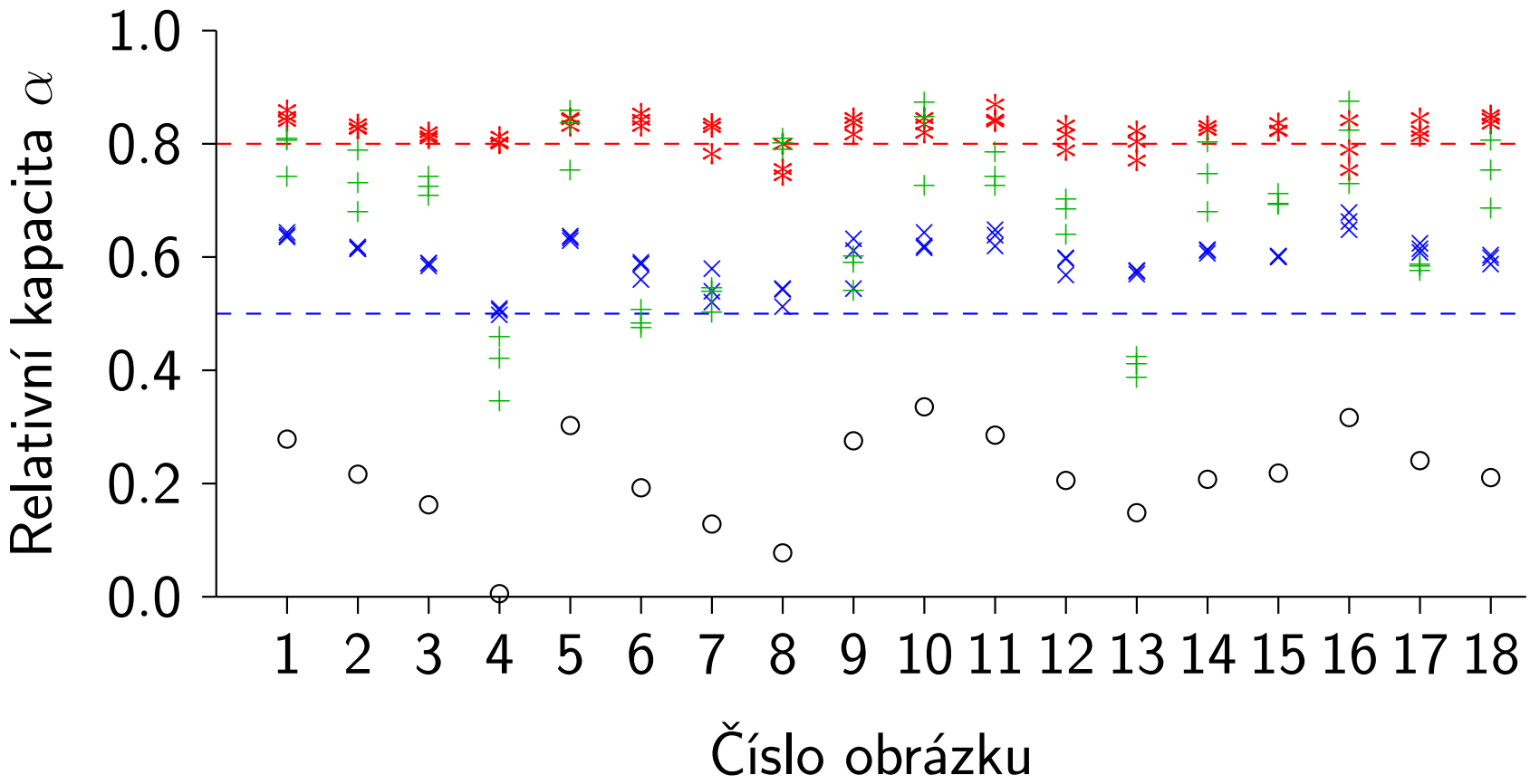
Po každém průchodu hlavním cyklem platí pro všechna $i \in I$ následující tři invarianty:

$$\begin{aligned} p(i) \text{ není definované} &\Leftrightarrow (E \cap (\mathbb{R} \times \{c_i\} \times C)) \neq \emptyset, \\ p(i) \text{ je definované} &\Leftrightarrow f(i) \text{ je definované}, \\ p(i) \text{ je definované} &\Rightarrow p(i) = 1 - p(f(i)). \end{aligned}$$

Na konci algoritmu platí:

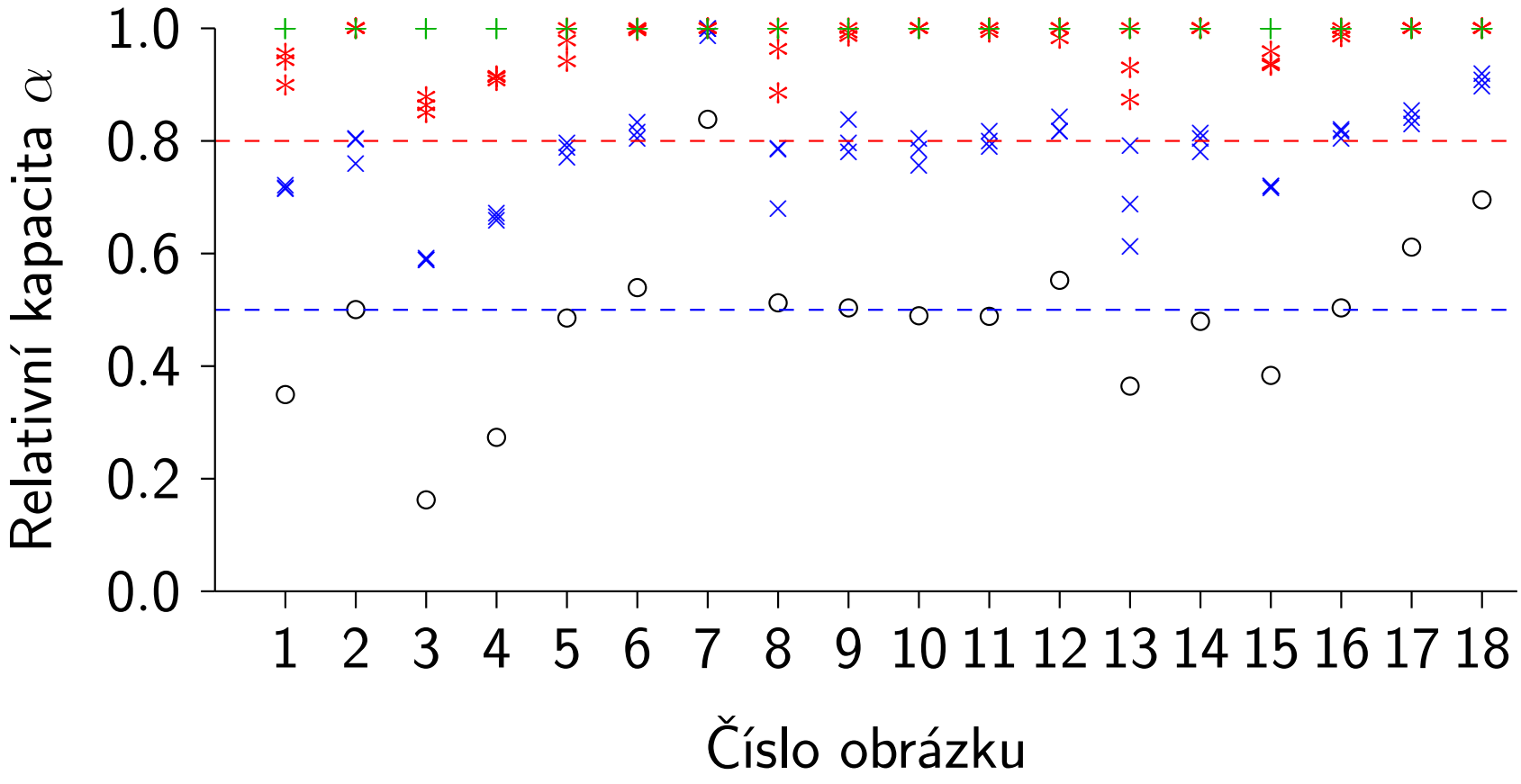
- $p(i) \in \{0, 1\}$ pro všechna $i \in I$. (Množina E je prázdná.)
- f je řádně definované zobrazení.
- f obrací paritu.
- Pro každé $i \in I$ je $\|c_i - c_{f(i)}\| = \min_{j \in I \setminus \{i\}} \|c_i - c_j\|$.

Útok na vkládání s optimálním přiřazením parity



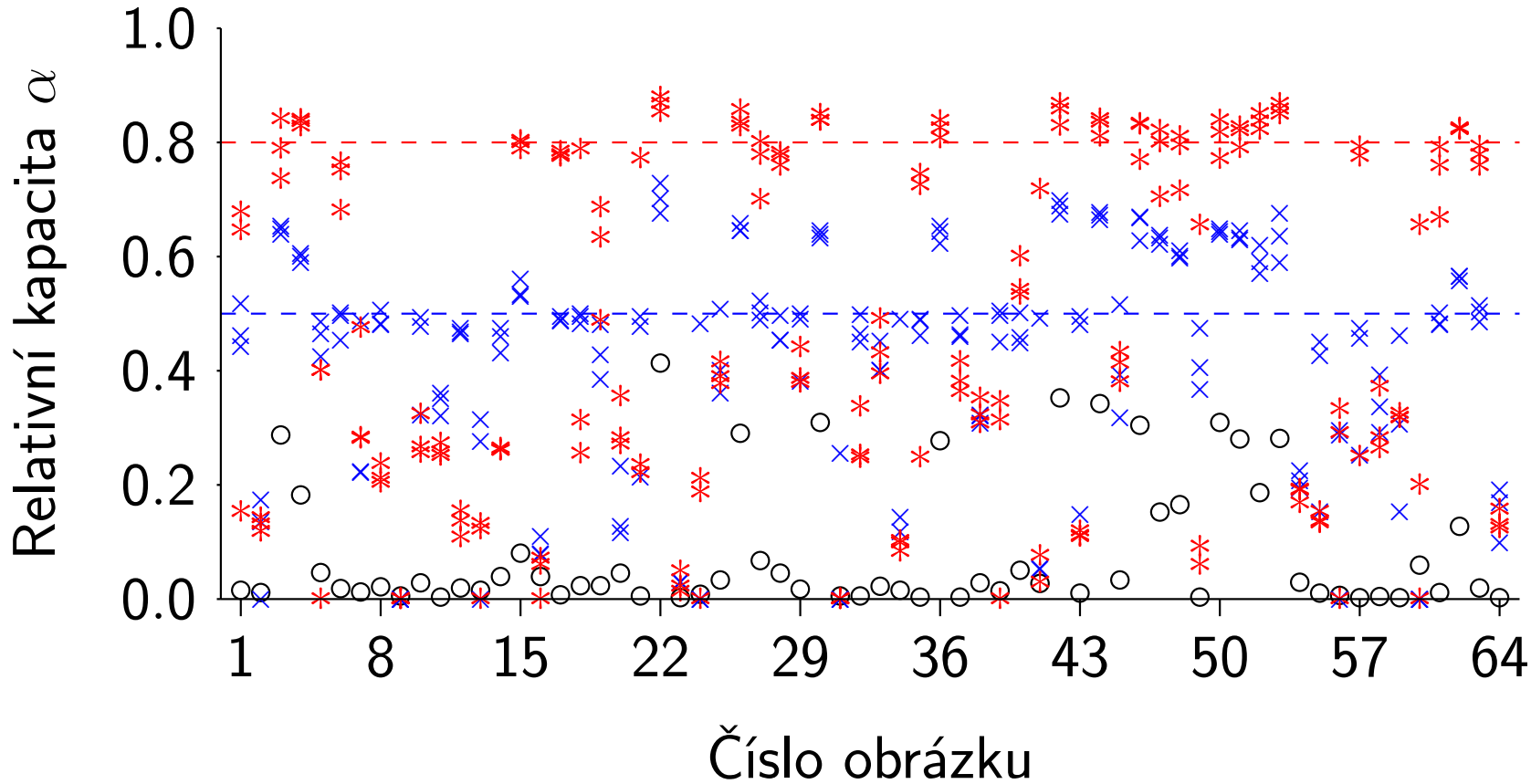
Graf zobrazující odhad relativní kapacity pro nosiče ($\circ$) a stegoobrázky s $\alpha = 0.5$ ($\times$), $\alpha = 0.8$ ($*$) a $\alpha = 1$ ($+$). Obrázky 512×512 s 256 barvami vytvořené pomocí ImageMagick z http://www4.comp.polyu.edu.hk/~cslzhang/CDM_Dataset.htm.

Útok na vkládání s optimálním přiřazením parity



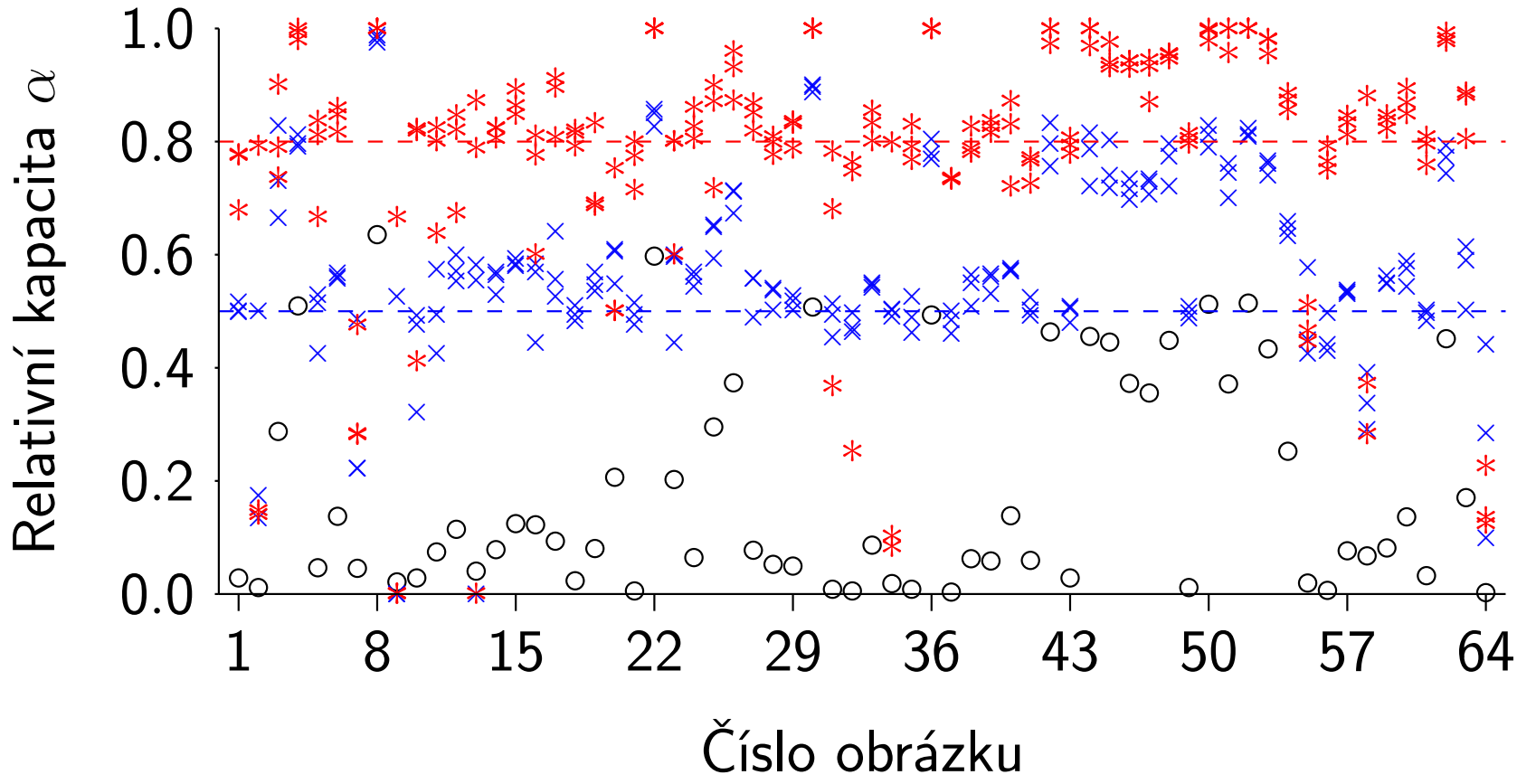
Tytéž obrázky, z histogramu byly vyřazeny barvy ležící v jádrech.

Útok na vkládání s optimálním přiřazením parity



Útok na obrázcích vytvořených ze sbírky http://decsai.ugr.es/cvg/dbimagenes/c512.php. V histogramu byly ponechány barvy ležící v jádrech.

Útok na vkládání s optimálním přiřazením parity



Tytéž obrázky, z histogramu byly vyřazeny barvy ležící v jádrech.

Vkládání při redukci hloubky barev

- Nosič je např. 24-bitový rastrový obrázek.
- Sestrojíme paletu a optimální přiřazení parity.
- Provedeme redukci hloubky barev s difuzí chyb.
- U pixelů nesoucích bit zprávy postup upravíme:
  - Při zaokrouhlování na nejbližší barvu se omezíme na množinu barev jejichž parita odpovídá bitu zprávy.

Adaptivní vkládání

Nápad:

- Nechceme vkládat do oblastí s uniformní barvou. (Přesvícené oblasti, nebe a jiné gradienty).
- Chceme vkládat do oblastí s vysokou texturou.

Příklad:

- Obrázek rozdělíme na bloky 3×3 pixely.
- Definujeme funkci t, která měří texturu bloku.
- Např. $t(B) =$ počet různých barev v B.
- Zvolíme prahovou hodnotu γ, např. $\gamma = 2$ nebo $\gamma = 3$.
- Do bloku B vkládáme, právě když $t(B) > \gamma$.

Adaptivní vkládání

Problém:

- Vkládání může texturu snížit.
- Příjemce nemá k dispozici nosič, aby mohl určit co byla oblast s vysokou či nízkou texturou.

Řešení:

- Poté co provedeme vkládání do bloku, změříme texturu znovu.
- Klesla-li textura pod prahovou hodnotu:
  - Blok odešleme se změnami.
  - Blok bude příjemcem ignorován.
  - Vložená informace je tedy ztracena.
  - Bity musíme vložit znovu do dalšího bloku.