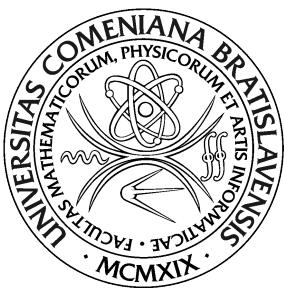


UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



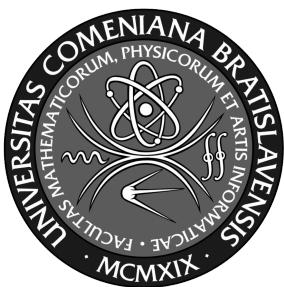
GRUPY AUTOMORFIZMOV LINEÁRNYCH KÓDOV

Diplomová práca

2022

Bc. Branislav Boráň

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



GRUPY AUTOMORFIZMOV LINEÁRNYCH KÓDOV

Diplomová práca

- Študijný program: Aplikovaná informatika
Študijný odbor: 2511 Aplikovaná informatika
Školiace pracovisko: Katedra algebry a geometrie
Školiteľ: doc. RNDr. Róbert Jajcay, DrSc.

Bratislava, 2022

Bc. Branislav Boráň



ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Branislav Boráň

Študijný program: aplikovaná informatika (Jednoodborové štúdium,
magisterský II. st., denná forma)

Študijný odbor: informatika

Typ záverečnej práce: diplomová

Jazyk záverečnej práce: anglický

Sekundárny jazyk: slovenský

Názov: Automorphism groups of linear codes and linear codes with prescribed automorphism groups

Grupy automorfizmov lineárnych kódov a lineárne kódy s predpisanou grupou automorfizmov

Anotácia: Lineárne kódy sú podpriestory konečnorozmerných vektorových priestorov nad konečnými poľami. Majú preto bohaté grupy automorfizmov, ktoré zároveň obsahujú množstvo informácií o uvažovanom kóde. Určenie úplnej grupy automorfizmov kódu je výpočtovo náročná úloha. Namiesto určenia grupy automorfizmov pre daný kód sa preto uvažuje obrátená úloha zostrojenia kódu s predpisanou grupou automorfizmov. Cieľom práce je preskúmať oba smery tejto interakcie.

Cieľ: Cieľom navrhanej problematiky je poskytnúť študentovi výpočtovo zložitý problém vyžadujúci dôkladné porozumenie štruktúry uvažovaných objektov ako aj programátorské a organizačné schopnosti.

Literatúra: R. Hill, A first course in coding theory, Oxford University Press, 1993
S. Roman, Coding and information theory, Springer, 1992
R. Jajcay, P. Potocnik and Stephen E. Wilson, Half-cyclic, dihedral and half-dihedral codes,
J. of Applied Mathematics and Computing 64 (2020), 691-708.

Kľúčové

slová: lineárny kód, grúpa automorfizmov, konečné pole

Vedúci: doc. RNDr. Róbert Jajcay, DrSc.

Katedra: FMFI.KAG - Katedra algebry a geometrie

Vedúci katedry: doc. RNDr. Pavel Chalmovianský, PhD.

Dátum zadania: 09.12.2020

Dátum schválenia: 10.12.2020

prof. RNDr. Roman Ďuríkovič, PhD.
garant študijného programu

študent

vedúci práce

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa.

.....
Bratislava, 2022

Bc. Branislav Boráň

Pod'akovanie

Chcel by som sa v prvom rade pod'akovať môjmu školiteľovi doc. RNDr. Róbertovi Jajcayovi, DrSc. za odbornú pomoc a usmernenia pri písaní tejto práce, za materiály, cenné rady, ktoré mi veľmi pomohli pri riešení tejto diplomovej práce. V neposlednom rade chcem tiež pod'akovať všetkým mojím kamarátom a celej mojej rodine za podporu počas môjho štúdia.

Abstrakt

Táto práca sa venuje problematike skúmaniau grúp automorfizmov lineárnych kódov ako aj lineárnym kódom s predpísanou grupou automorfizmov. Cieľom práce je preskúmať oba smery tejto interakcie a poskytnúť študentovi výpočtovo zložitý problém vyžadujúci dôkladné porozumenie štruktúry uvažovaných objektov ako aj programátorské a organizačné schopnosti. Riešenie sa zakladá na využití knižnice Sage dvoma spôsobmi. Prvé riešenie bude implementované v online aplikácii CoCalc vo forme Sage skriptov. Druhé riešenie bude implementované ako konzolová aplikácia v programovacom jazyku Python, kde bude knižnica Sage naimportovaná.

Kľúčové slová: grúpa automorfizmov, perfektné kódy, klietky

Abstract

This thesis deals with the problem of determining the groups of automorphisms of linear codes as well as with linear codes a prescribed group of automorphisms. The aim of the work is to examine both directions of this interaction and to provide the student a computationally complex problem requiring a thorough understanding of the structure of the considered objects as well as programming and organizational skills. The solution is based on the use of the Sage library in two ways. The first solution will be implemented in the online application CoCalc in the form of Sage scripts. The second solution will be implemented as a console application in the Python programming language, where the Sage library will be imported.

Keywords: automorphism group, perfect codes, cages

Obsah

1	Úvod	1
2	Motivácia	3
3	Analýza problému	4
3.1	Lineárny kód	4
3.1.1	Generujúca matica lineárneho kódu	5
3.1.2	Kontrolná matica lineárneho kódu	5
3.1.3	Dĺžka lineárneho kódu	6
3.1.4	Počet slov v kóde a Optimálny kód	6
3.1.5	Minimálna Hammingova vzdialenosť v kóde	8
3.1.6	Informačný pomer	9
3.1.7	Perfektné lineárne kódy	9
3.1.8	LDPC kódy	10
3.2	Grafová reprezentácia LDPC kódov	11
3.2.1	Graf vzdialenosťi	11
3.2.2	Veľkosť obvodu grafu	13
3.2.3	Automorfizmus grafu	13
3.2.4	Konštrukcia LDPC kódov pomocou klietok	13

4 Existujúce riešenia	17
4.1 Dizertačná práca Low-Density Parity-Check Codes: Construction and Implementation [Mal07]	17
4.2 Článok Low Complexity, High Performance LDPC Codes Based on Defected Fullerene Graphs [Gos12]	18
4.3 Diplomová práca On The Automorphism Groups of Some Linear Codes [Nas19]	18
4.4 Článok Dynamic Cage Survey [EJ13]	18
5 Návrh riešenia	19
5.1 Generovanie klietok a rekordných grafov	20
5.1.1 Sage grafy ako vstupné dátá	20
5.1.2 Zoznam susedností ako vstupné dátá	21
5.1.3 Vlastný návrh generovania	21
5.2 Validácia získaných klietok alebo rekordných grafov	22
5.3 Získavanie údajov z klietok alebo rekordných grafov	22
5.3.1 Počty vrcholov, hrán a automorfizmov, informačný po- mer	23
5.3.2 Incidenčné matice	23
5.4 Lineárny kód a generujúca matica získaná z incidenčnej matice klietky	24
5.4.1 Minimálna kódová vzdialenosť, maximálny počet slov a grupa automorfizmov	24
5.4.2 Porovnanie s optimálnymi kódmi	25
5.4.3 Porovnanie s perfektnými kódmi	25
5.5 Validácia lineárneho kódu	25
6 Riešenie	27

6.1	Generovanie klietky, rekordného grafu a generujúcej matice	27
6.1.1	Dôležité funkcie	28
6.1.2	Cage(3,5) - Petersenov graf	34
6.1.3	Cage(3,6) - Heawoodov graf	36
6.1.4	Cage(3,7) - McGeeho graf	38
6.1.5	Cage(3,8) - Tutteho-Coxeterov graf	40
6.1.6	Cage(3,10) - Balabanov graf	42
6.1.7	Cage(3,11) - Balabanov graf	43
6.1.8	Rec(3,14)	45
6.1.9	Rec(3,16)	46
6.1.10	Rec(3,17)	47
6.1.11	Rec(3,18)	48
6.1.12	Rec(3,20)	49
6.1.13	Rec(3,23)	50
6.1.14	Rec(3,25)	51
6.1.15	Cage(4,5) - Robertsonov graf	52
6.1.16	Cage(4,7)	54
6.1.17	Cage(4,9)	55
6.1.18	Cage(4,10)	57
6.1.19	Cage(5,10)	58
6.1.20	Cage(6,4)	59
6.1.21	Cage(7,5)	61
6.1.22	Cage(7,7)	62
6.1.23	Cage(7,8)	63
6.1.24	Rec(10,5)	64
6.1.25	Rec(11,5)	65
6.1.26	Rec(12,5)	67

<i>OBSAH</i>	xi
--------------	----

6.1.27 Rec(13,5)	68
6.2 Výsledky	69
6.2.1 Zhrnutie výsledkov	69
6.2.2 Vyhodnotenie výsledkov	70
7 Záver	72

Kapitola 1

Úvod

V našej práci využívame na konštrukciu kódov klietky, prípadne rekordné grafy, ktoré sú špecialny typ grafu vzdialenosť. Tento graf nám v maticovej reprezentácii poskytuje kontrolnú maticu lineárneho kódu, na základe ktorej sa vieme dopracovať k lineárnym kódom. Navrhujeme riešenie dvoma spôsobmi. Prvé riešenie je v online aplikácii CoCalc vo forme Sage skriptov. Druhé riešenie pozostáva z konzolovej aplikácie v programovacom jazyku Python, kde je potrebné si knižnicu Sage naimportovať. Výsledná aplikácia je spustiteľná len po inštalácii Sage v Sage konzole. Ako prvý krok je potrebné si vygenerovať klietku alebo rekordný graf a to buď využitím vopred naimplementovaných grafov zo Sage (využijeme triedu

sage.graphs.graph_generators.GraphGenerators [The+20]), vytvorením parsera na spracovanie zoznamu susedností alebo si navrhнемe klietku vlastným spôsobom na základe informácií, ktoré o nej vieme. Z klietok respektíve rekordných grafov vieme následne určiť počet vrcholov (využijeme funkciu *vertices()* [**sagemath**]), hrán (pomocou funkcie *edges()* [**sagemath**]), automorfizmov (pomocou funkcie *automorphism_group()* [**sagemath**]), informačný pomer a incidenčné matice (pomocou funkcie *incidence_matrix()*

[**sagemath**]). Z incidenčnej matice vieme určiť lineárny kód (pomocou funkcie

codes.from_parity_check_matrix(H) [**sagemath**]), generujúcu maticu (pomocou funkcie *systematic_generator_matrix()* [**sagemath**]), minimálnu kódovú vzdialenosť (pomocou funkcie *minimum_distance()* [**sagemath**]), počet slov a počet automorfizmov (pomocou funkcií *permutation_automorphism_group()* a *order()* [**sagemath**]). Lineárne kódy vieme porovnať s optimálnymi a perfektnými kódmi.

Kapitola 2

Motivácia

Lineárne kódy sú podpriestory konečnorozmerných vektorových priestorov nad konečnými poľami. S lineárnymi (LDPC aj MDPC) kódmi som sa stretol už vo svojej bakalárskej práci [Bor18], kde som skúmal hustotu inverzií riedkych cyklických matíc v McElieceovom kryptosystéme. Predmetom skúmania bola hustota inverzných matíc k riedkym cyklickým maticiam. Zistili sme, že matica, ktorá predstavovala blok generujúcej matice, mala v niektorých prípadoch nízku hustotu a mohla by znamenať bezpečnostné riziko pre QC-MDPC. V tejto práci sme voľne nadviazali na tému lineárnych kódov, ale tentokrát sa špecializujeme na súvis so špeciálnym typom grafu vzdialenosť, s klietkami respektíve rekordnými grafmi. Snažíme sa využiť tie najznámejšie grafy a získať z nich lineárne kódy, ktoré na základe poznatkov o známych perfektných kódoch a optimálnych kódoch vieme vyhodnotiť. Tieto lineárne kódy majú bohaté grupy automorfizmov, ktoré pomáhajú pri dekódovaní a obsahujú množstvo informácií o uvažovanom kóde. Určenie úplnej gruhy automorfizmov kódu je výpočtovo náročná úloha. Namiesto určenia gruhy automorfizmov pre daný kód sa preto uvažuje obrátená úloha zostrojenia lineárneho kódu s predpísanou grupou automorfizmov.

Kapitola 3

Analýza problému

3.1 Lineárny kód

Nech $F(2)$ je priestor n -rozmerných vektorov. Lineárny kód $C(n,r)$ je r -rozmerný lineárny podpriestor priestoru $F(2)$. Množinu všetkých usporiadaných n -tíc $F(2)$ označujeme $V(n,2)$ a jej prvky budeme nazývať vektory [Hil88]. Lineárny podpriestor rozmeru r obsahuje práve r lineárne nezávislých vektorov. Ak by sme zobraли r takých vektorov, potom tieto vektory generujú daný r -rozmerný podpriestor a hovoríme, že tvoria bázu tohto podpriestoru [Mis+13] [HP03].

Veta č.1

Predpokladajme, že $\{v_1, v_2, \dots, v_r\}$ je bázou podpriestoru $C(n,r)$ z $V(n,2)$. Potom každý vektor $C(n,r)$ môže byť vyjadrený jednoznačne ako lineárna kombinácia základných vektorov a $C(n,r)$ obsahuje presne 2^r vektorov [Hil88].

Ak akékoľvek dve bázy podpriestoru $C(n,r)$ obsahujú rovnaký počet vektorov r , kde $|C(n,r)| = 2^r$, potom číslo r nazývame dimensiou podpriestoru

$C(n,r)$ a označujeme ju $\dim(C)$. V prípade, že $C(n,r)$ je r -dimenzionálny podpriestor z $V(n,2)$, potom lineárny kód $C(n,r)$ je nazývaný $[n,r]$ -kód, avšak v prípade, že by sme chceli špecifikovať minimálnu vzdialenosť lineárneho kódu d , tak lineárny kód označujeme ako $[n,r,d]$ -kód. Takýto binárny $[n,r,d]$ -kód môže byť chápaný ako $(n,2^r,d)$ -kód [Veta č.1] a budeme ho označovať Lineárny kód $C(n,m,d)$, kde m je maximálny počet slov a d je minimálna kódová vzdialenosť [Hil88].

Vo všeobecnom kóde platí vlastnosť, ktorá hovorí, že ak súčet každých dvoch kódových slov $\text{mod } 2$ je kódové slovo, tak uvažovaný binárny kód je lineárny a vieme nájsť generujúcu maticu lineárneho kódu G [Mal07] [Hil88].

3.1.1 Generujúca matica lineárneho kódu

Generujúca matica lineárneho kódu G je zostrojená z bázy lineárneho kódu $C(n,r)$. Riadky matice predstavujú prvky bázy a zároveň predstavujú lineárne nezávislé vektory dĺžky n [Mis+13] [HP03]. Generujúcu maticu G si môžeme zadefinovať nasledovne. Nech je vektor \vec{s} vstup, teda nekódované slovo a vektor \vec{x} je výstup, teda kódované slovo, potom platí:

$$C(n, r) = \{\vec{s} \times G : \vec{s} \in F(2)^r\}, \quad \vec{x} = \vec{s} \times G \quad (3.1)$$

3.1.2 Kontrolná matica lineárneho kódu

V r -rozmersnom linearnom kóde $C(n,r)$ v $F(2)$ potom existuje $n - r$ lineárne nezávislých vektorov \vec{x} takých, že každé kódové slovo je kolmé na všetky tieto vektory. Keď týchto $n - r$ vektorov zoberieme ako riadky matice, dostaneme kontrolnú maticu lineárneho kódu H [Mis+13] [HP03]. Ľubovoľný vektor \vec{x}

je kódovým slovom práve vtedy, ak platí:

$$C(n, r) = \{\vec{x} \in F(2) : H \times \vec{x}^T = 0\} \quad (3.2)$$

3.1.3 Dĺžka lineárneho kódu

Dĺžka lineárneho kódu n predstavuje dĺžku slov v kóde. V kontrolnej matici H predstavuje počet lineárne nezávislých riadkov a v Generujúcej matici G predstavuje počet stĺpcov [Mis+13]. V prípade, že uvažujeme bázu z $V(n, 2)$, ktorá má n vektorov, tak platí [Hil88]:

$$n = \dim(V(n, 2)) \quad (3.3)$$

3.1.4 Počet slov v kóde a Optimálny kód

Počet slov v kóde m je počet takých kódových slov, ktoré obsahuje lineárny kód $C(n, m, d)$. Nech parameter r predstavuje počet lineárne nezávislých riadkov generujúcej matice G , potom z vety č.1 vyplýva: [Hil88]:

$$m = |C(n, r)| = 2^r \quad (3.4)$$

Veta č.2 (Optimálny kód)

Uvažujme abecedu A dĺžky 2 a fixné parametre n a d , potom lineárny $[n, r, d]$ -kód, pre ktorý platí $2^r = A_2(n, d)$ sa nazýva optimálny lineárny kód $C(n, m, d)$ [Hil88]:

$$A_2(n, d) = \max\{m = 2^r \mid \exists[n, r, d] \in F(2)\} \quad (3.5)$$

V nasledujúcich tabuľkách si zobrazíme známe $A_2(n, d)$ pre niektoré filxné

dĺžky lineárneho kódu n a minimálne vzdialenosťi v kóde d [Hil88] [Bro18]:

n	d = 3	d = 4	d = 5	d = 6
5	4	-	2	-
6	8	4	2	2
7	16	8	2	2
8	20	16	4	2
9	40	20	6	4
10	72-79	40	12	6
11	144-158	72	24	12
12	256	144	32	24
13	512	256	64	32
14	1024	512	128	64
15	2048	1024	256	128
16	2816-3276	2048	258-340	256
17	5632-6552	2816-3276	512-673	258-340
18	10496-13104	5632-6552	1024-1237	512-673
19	20480-26168	10496-13104	2048-2279	1024-1237
20	40960-43688	20480-26168	2560-4096	2048-2279
21	81920-87333	40960-43688	4096-6941	2560-4096
22	163840-172361	81920-87333	8192-13674	4096-6941
23	327680-344308	163840-172361	16384-24106	8192-13674
24	$2^{19} - 599184$	327680-344308	17920-47538	16384-24106
25	$2^{20} - 1198368$	$2^{19} - 599184$	32768-84260	17920-47538
26	$2^{21} - 2396736$	$2^{20} - 1198368$	65536-157285	32768-84260
27	$2^{22} - 4792950$	$2^{21} - 2396736$	131072-291269	65536-157285
28	-	$2^{22} - 4792950$	-	131072-291269

Tabuľka 3.1: Tabuľka známych $A_2(n, d)$ č.1

n	d = 7	d = 8	d = 10	d = 12	d = 14	d = 16
5	1	-	-	-	-	-
6	1	1	1	1	1	1
7	2	1	1	1	1	1
8	2	2	1	1	1	1
9	2	2	1	1	1	1
10	2	2	2	1	1	1
11	4	2	2	1	1	1
12	4	4	2	2	1	1
13	8	4	2	2	1	1
14	16	8	2	2	2	1
15	32	16	4	2	2	1
16	36-37	32	4	2	2	2
17	64	36	6	2	2	2
18	128	64	10	4	2	2
19	256	128	20	4	2	2
20	512	256	40	6	2	2
21	1024	512	42-47	8	4	2
22	2048	1024	64-84	12	4	2
23	4096	2048	80-150	24	4	2
24	4096-5421	4096	136-268	48	6	4
25	4104-9275	4096-5421	192-466	52-55	8	4
26	8192- 17099	4104-9275	384-836	64-96	14	4
27	16384- 32151	8192- 17099	512-1585	128-169	28	6
28	-	16384- 32151	1024-2817	178-288	56	8

Tabuľka 3.2: Tabuľka známych $A_2(n, d)$ č.2

3.1.5 Minimálna Hammingova vzdialenosť v kóde

Uvažujme vektory \vec{x} a \vec{y} ako kódové slová. Minimálna Hammingova vzdialenosť vektorov $d(\vec{x}, \vec{y})$, pre ktoré platí, že $\vec{x} \in F_2$ a $\vec{y} \in F_2$ je počet ko-

ordinátov, na ktorých sa vektory \vec{x} a \vec{y} líšia [Mis+13] [HP03]. Minimálna Hammingova vzdialenosť v kóde d je taká najmenšia vzdialenosť v lineárnom kóde $C(n,m,d)$, ktorá je definovaná ako najmenšia vzdialenosť zo všetkých vzdialostí medzi kódovými slovami, pričom platí [Hil88]:

$$d = \min\{d(\vec{x}, \vec{y}) | \vec{x}, \vec{y} \in C(n, m, d), \vec{x} \neq \vec{y}\} \quad (3.6)$$

Je dôležitým parametrom lineárneho kódu $C(n,m,d)$ z hľadiska opravy chýb [Hil88].

3.1.6 Informačný pomer

Informačný pomer R (angl. Rate) udáva počet informácií alebo bitov nad celkovým počtom prenesených bitov. Nech $|cols|$ predstavuje počet stĺpcov kontrolnej matice H a $|rows|$ predstavuje počet riadkov kontrolnej matice H. Potom R je vyjadrený nasledujúcim vzťahom [Mal07]:

$$R = \frac{|cols| - |rows|}{|cols|} \quad (3.7)$$

3.1.7 Perfektné lineárne kódy

Veta č.3 (Sphere-packing ohraničenie)

Nech t je parameter, pre ktorý platí, že každé dve sféry polomeru t centrovane na odlišné kódové slová majú prázdný prienik. Pomocou parametra t môžeme vyjadriť minimálnu vzdialenosť v kóde d nasledovne [Hil88]:

$$d = 2t + 1 \quad (3.8)$$

Predpokladajme, že Lineárny kód $C(n,m,d)$ je binárny $(n, m, 2t + 1)$ -kód. Sphere-packing ohraničenie je dané vzťahom:

$$m \left\{ \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{t} \right\} \leq 2^n \quad (3.9)$$

Počet všetkých vektorov v m sférach s polomerom t centrovaných na m kódových slov je daný ľavou stranou rovnice. Sphere-packing ohraničenie platí vždy vtedy, ak je ľavá strana rovnice menšia nanajvýš rovnajúca sa 2^n . Toto číslo vyjadruje celkový počet vektorov v $F(2)$. V prípade, že sú obe strany rovnice v rovnosti, tak hovoríme, že Lineárny kód $C(n,m,d)$ je perfektný [Hil88].

3.1.8 LDPC kódy

LDPC kódy sú lineárne samoopravné kódy, ktoré umožňujú prenos dát rýchlosťou blízkou kapacite kanálu a zároveň pre ne existujú vysoko účinné dekódovacie algoritmy. Tieto kódy majú veľmi riedku kontrolnú maticu H , pomocou ktorej sa dajú opraviť chyby v kódových slovách. Obsahuje menej ako 1% jednotiek [Mis+13]. Hlavnou nevýhodou väčšiny LDPC kódov je vysoká časová náročnosť ich kódovacieho algoritmu. Výhodou je paralelizmus pri dekódovaní a jednoduché výpočtové operácie. Dekódovacie výpočty sú rozdelené do 2 množín uzlov a to do kontrolných uzlov a premenných uzlov. Uzol na jednej strane je spojený s uzlom na druhej strane, čo umožňuje paralelné výpočty na každej strane [Mal07]. Téme LDPC kódov som sa okrajovo venoval aj vo svojej bakalárskej práci, v ktorej som skúmal hustotu inverzií riedkych cyklických matíc. Zameral som sa však na QC-MDPC McElieceov

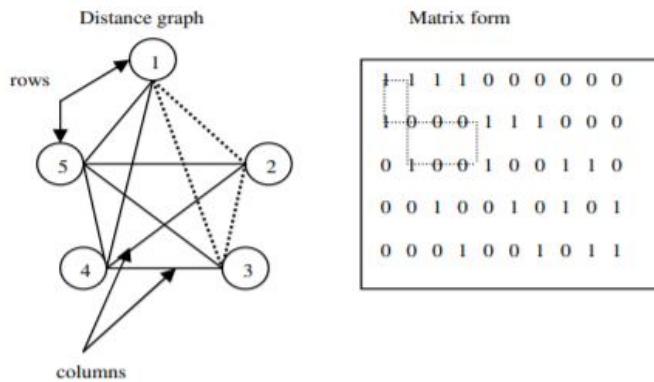
kryptosystém. Rozdiel medzi QC-MDPC kódmi a LDPC je v hustote kontrolnej matice, ktorá môže byť pre QC-MDPC kódy o niečo hustejšia ako pri LDPC kódoch (obsahuje menej ako 2% jednotiek) [Bor18].

3.2 Grafová reprezentácia LDPC kódov

3.2.1 Graf vzdialenosťi

Graf vzdialenosťi je spojený graf, ktorý má daný počet vrcholov $|ver|$, stupeň vrcholov k , v ktorom minimálna dĺžka cyklu medzi vrcholmi a hranami je g [Mal07]. Vrcholy grafu vzdialenosťi reprezentujú riadky kontrolnej matice H a počet takýchto vrcholov v grafe vzdialenosťi zodpovedá počtu riadkov v kontrolnej matici H . Hrany grafu vzdialenosťi reprezentujú stĺpce kontrolnej matice H a počet takýchto hrán v grafe vzdialenosťi zodpovedá počtu stĺpcov v kontrolnej matici H . Stĺpec kontrolnej matice H potom predstavuje takú množinu hrán, ktorá formuje kompletný graf vzdialenosťi medzi vrcholmi spojenými v stĺpci [Mal07].

Kontrolná matica lineárneho kódu H teda môže byť zobrazená ako graf vzdialenosťi. Nasledujúci obrázok ilustruje vzťah medzi kontrolnou maticou LDPC kódu a grafom vzdialenosťi z ktorého je kontrolná matica H odvodená [Mal07]:



Obr. 3.1: Vzťah medzi grafom a kontrolnou maticou [Mal07]

Graf vzdialenosť je formovaný cestami, ktoré sa skladajú buď z hrán alebo vrcholov. Cestu ktorá sa začína a končí tým istým vrcholom alebo cestu, ktorá sa začína a končí tou istou hranou budeme nazývať obvod grafu vzdialenosť. V kontrolnej matici H je obvod reprezentovaný uzavretou cestou, ktorá sa skladá z pospájaných prvkov hodnoty 1 po vertikálnej a horizontálnej línií tak, ako je zobrazené na obrázku. Môžeme si všimnúť, že prvý riadok (vrchol 1) je spojený s druhým riadkom (vrchol 2) na prvej pozícii vertikálne, pretože oba riadky majú na začiatku prvok 1. V druhom riadku hľadáme horizontálne najbližší prvok hodnoty 1, ktorý by sme opäť mohli spojiť vertikálne s ďalším prvkom 1, nachádzajúcim sa v inom riadku. V prípade, že dokážeme vertikálne spojiť prvok 1 v ľubovoľnom riadku opäť s prvým riadkom, cesta sa uzavrie a dostávame obvod kontrolnej matice H . Východiskový riadok nemusí byť nutne prvý, ale aby bol obvod kompletný, je nutné východiskovým riadkom skončiť. Jednotlivé obvody v grafe vzdialenosť korešpondujú s obvodmi v kontrolnej matici H [Mal07].

3.2.2 Veľkosť obvodu grafu

Veľkosť obvodu grafu g ovplyvňuje rýchlosť dekódovania LDPC kódu. V grafovej reprezentácii LDPC kódu sa jedná o najmenší cyklus v grafe. Jeho dĺžku vypočítame buď pomocou vrcholov alebo pomocou hrán. Ako sme spomínali, jednotlivé obvody v grafe vzdialenosť korešpondujú s obvodmi v kontrolnej matici H . V kontrolnej matici H kódu je dĺžka obvodu $2g$, pretože cyklus medzi riadkami a stĺpcami je vedený nielen vertikálne ale aj horizontálne, pričom do veľkosti obvodu sa započítavaju všetky uvažované prepojenia s prvkami hodnoty 1. Z toho vyplýva, že cyklus grafu reprezentuje iba polovicu maticového kódu [Mal07].

3.2.3 Automorfizmus grafu

Automorfizmus grafu je permutácia ϕ všetkých vrcholov grafu, ktorá zachováva jeho štruktúru takým spôsobom, že akékoľvek 2 vrcholy U a V susedia iba vtedy a len vtedy ak platí, že $\phi(U)$ susedí s $\phi(V)$ [EJ13]. Množina všetkých automorfizmov grafu tvorí grupu automorfizmov $\text{AutGroup}(\text{Cage})$ [EJ13].

3.2.4 Konštrukcia LDPC kódov pomocou klietok

Na konštrukciu LDPC kódov môžme využiť grafy vzdialenosťi. Tieto grafy delíme na regulárne s vrcholmi rovnakého stupňa k (napr. Moorové grafy) a neregulárne s vrcholmi rôznych stupňov k [Mal07]. V našej práci sa budeme venovať regulárnym grafom vzdialenosťi. **Klietka Cage(k,g)** je k -regulárny graf obvodu g s počtom vrcholov $|ver|$ väčším nanajvýš rovnajúcim sa dolnému ohraničeniu počtu vrcholov, tiež známym ako **Moorové ohraničenie pre klietku $M(k,g)$** [EJ13]. Výpočet Moorovho ohraničenia sa lísi podľa

toho, či je obvod g uvažovanej klietky $\text{Cage}(k,g)$ párny alebo nepárny:

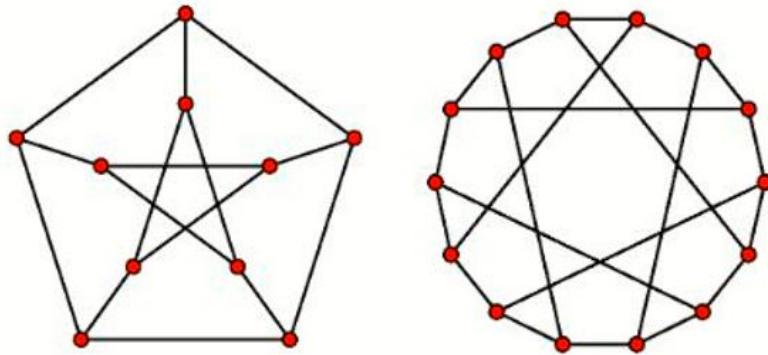
nepárny obvod:

$$M(k, g) \leq 1 + \sum_{i=0}^{(g-3)/2} k(k-1)^i = \frac{k(k-1)^{(g-1)/2} - 2}{k-2} \quad (3.10)$$

párny obvod:

$$M(k, g) \leq 2 \sum_{i=0}^{(g-2)/2} k(k-1)^i = \frac{2(k-1)^{g/2} - 2}{k-2} \quad (3.11)$$

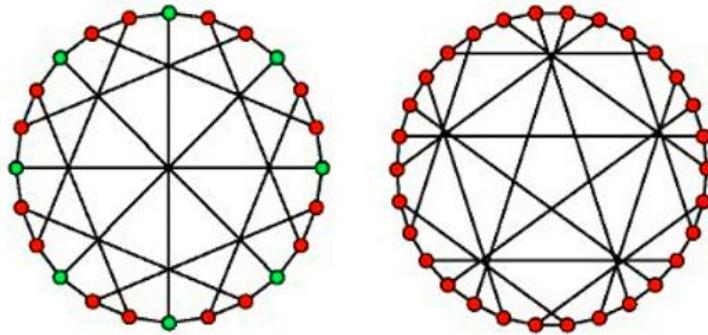
Aj keď neexistuje jednotná konštrukcia klietok, existuje niekoľko známych klietok pre stupeň vrchola k a obvod g [EJ13]. Ukážeme si niektoré z nich:



Obr. 3.2: Petersenov graf (vľavo) a Heawoodov graf (vpravo) [EJ13]

Petersenov graf predstavuje klietku $cage(3, 5)$ a počet vrecholov $|\text{ver}| = 10$.

Heawoodov graf je klietkou $cage(3, 6)$, počet vrcholov $|\text{ver}| = 14$ a počet automorfizmov $|\text{AutGroup}(\text{Cage})| = 336$ [EJ13].



Obr. 3.3: McGeeho graf (vľavo) a Tutteov-Coxeterov graf (vpravo)[EJ13]

McGeeho graf znázorňuje klietku $cage(3, 7)$, počet vrcholov $|ver|= 24$ a počet automorfizmov $|\text{AutGroup}(\text{Cage})|= 32$ [EJ13].

Tutteho-Coxterov graf predstavuje klietku $cage(3, 8)$, počet vrcholov $|ver|= 30$ a počet automorfizmov $|\text{AutGroup}(\text{Cage})|= 1440$ [EJ13].

Balabanov graf znázorňuje klietku $cage(3, 11)$, počet vrcholov $|ver|= 112$ a počet automorfizmov $|\text{AutGroup}(\text{Cage})|= 64$ [EJ13].

Bensonov graf je klietkou $cage(3, 12)$, počet vrcholov $|ver|= 126$ a počet automorfizmov $|\text{AutGroup}(\text{Cage})|= 12096$ [EJ13].

Ďalšie grafy:

Robertsonov graf predstavuje klietku $cage(4, 5)$

Exoo, McKay a Nadonov graf znázorňujú klietku $cage(4, 7)$

$cage(5, 5)$ je skupina grafov, ktoré majú počty automorfizmov 20, 30 a 120

Hoffmanov-Singletonov graf je klietkou $cage(7, 5)$

O'Keefe a Wongov graf predstavujú klietku $cage(7, 6)$ [EJ13]

Doposiaľ sme uvažovali známe klietky Cage(k,g). Jednalo sa o grafy, ktoré

rých počty vrcholov $|ver|$ sú preukázateľne najmenšie a také aj ostatné. Okrem klietok poznáme ešte **Rekordné grafy $Rec(k,g)$** . Rekordný graf $Rec(k,g)$ je graf, ktorého počet vrcholov $|ver|$ je najmenší v súčasnosti známy. Hoci niektoré z týchto grafov môžu byť v skutočnosti klietky, väčšina bude s najväčšou pravdepodobnosťou nakoniec nahradená menšími grafmi. [EJ13] V nasledujúcej tabuľke zobrazujeme rekordné grafy $Rec(k,g)$ so známym počtom vrcholov a automorfizmov:

Rec(k,g)	ver	AutGroup(Cage)
Rec(3,13)	272	-
Rec(3,14)	384	-
Rec(3,15)	620	14880
Rec(3,16)	960	96
Rec(3,17)	544	2176
Rec(3,18)	2560	640
Rec(3,19)	4324	51888
Rec(3,20)	5376	2688
Rec(3,21)	16028	-
Rec(3,22)	16206	-
Rec(3,23)	49326	-
Rec(3,24)	49608	-
Rec(3,25)	108906	-
Rec(8,5)	80	-
Rec(9,5)	96	-
Rec(10,5)	124	-
Rec(11,5)	154	-
Rec(12,5)	203	-
Rec(13,5)	230	-

Tabuľka 3.3: Tabuľka známych $Rec(k,g)$

Kapitola 4

Existujúce riešenia

Cieľom tejto práce je zestrojiť lineárne kódy $C(n,m,d)$ s predpísanými grupami automorfizmov ako aj určiť úplnú grupu automorfizmov pre daný lineárny kód $C(n,m,d)$ a preskúmať oba smery tejto interakcie. Pri zstrojovaní $C(n,m,d)$ vychádzame zo špeciálneho typu grafu vzdialenosťi, klietky Cage(k,g) prípadne z rekordného grafu Rec(k,g). Pri vyhľadávaní a skúmaní podobných riešení nášho problému sme našli niekoľko riešení, ktoré riešia časť nášho skúmaného problému, čo nám umožňuje nadviazať na ich riešenia.

4.1 Dizertačná práca Low-Density Parity-Check Codes: Construction and Implementation [Mal07]

V práci autor popisuje graf vzdialenosťi a spôsob akým je možné z neho získať kontrolnú maticu H . Práca uvádza okrajovo aj typ grafu vzdialenosťi klietky Cage(k,g). Pre uvažované klietky Cage(k,g) vedia vypočítať napríklad aj počet vrcholov $|ver|$, rozmer kontrolnej matice H , veľkosť obvodu g a

informačný pomer R . Všetky tieto údaje môžme využiť aj v našej práci.

4.2 Článok Low Complexity, High Performance LDPC Codes Based on Defected Fullerene Graphs [Gos12]

V článku autori popisujú známe klietky Cage(k,g) stupňa 3 (kubické klietky). Z kubických klietok skúmajú napríklad informačný pomer R , rozmer kontrolnej matice H , ktoré môžme využiť v našej práci. Takisto skúmajú čas potrebný na kódovanie v sekundách a porovnávajú klietky s Fulerénovými grafmi.

4.3 Diplomová práca On The Automorphism Groups of Some Linear Codes [Nas19]

V práci autor popisuje grupy automorfizmov $\text{AutGroup}(C)$ niektorých lineárnych kódov. Okrem iných kódov skúma aj práve binárne kódy.

4.4 Článok Dynamic Cage Survey [EJ13]

V článku autori detailne popisujú a skúmajú známe klietky Cage(k,g) a rekordné grafy Rec(k,g). Takisto uvádzajú pre tieto štruktúry napríklad počet vrcholov $|ver|$, počty automorfizmov $|\text{AutGroup}(\text{Cage})|$, ktoré môžme využiť v našej práci.

Kapitola 5

Návrh riešenia

Problematiku riešime v programe **Sage** [The+20], ktorý je založený na programovacom jazyku **Python**. Ponúka veľké množstvo vopred naimplementovaných funkcií pre prácu s grafmi a linearnymi kódmi $C(n,m,d)$. Využili sme online aplikáciu **CoCalc** [Sag20], ktorá nám umožňuje vytvárať Sage skripty priamo na internete ale aj aplikáciu **SageMath 9.2**, ktorá je voľne dostupná na stiahnutie, inštaláciu a podporuje aj skripty a projekty v jazyku Python. CoCalc prevádzkuje prostredie Ubuntu Linux, s ktorým je možné komunikovať cez terminál.



Obr. 5.1: Sage a Cocalc[sagemath][Sag20]

5.1 Generovanie klietok a rekordných grafov

Existuje viacero možností ako vygenerovať klietku $\text{Cage}(k,g)$ alebo rekordný graf $\text{Rec}(k,g)$ v programe Sage. V našej práci uvažujeme 3 spôsoby generovania $\text{Cage}(k,g)$ alebo $\text{Rec}(k,g)$. Prvý spôsob je **využietie grafov**, ktoré Sage ponúka ako vopred naimplementované grafy. Druhý spôsob sa opiera o naprogramovanie vlastného parsera, ktorý dokáže spracovať vstupné textové súbory **zoznamu susednosti** a z nich vytvoriť $\text{Cage}(k,g)$ alebo $\text{Rec}(k,g)$. Tretí spôsob spočíva vo **využití známych údajov**, ktoré sú pre danú klietku $\text{Cage}(k,g)$ všeobecne známe (počet vrcholov $|ver|$, typ grafu a podobne) a navrhnutie vlastného postupu, ktorý bude viest' k vygenerovaniu $\text{Cage}(k,g)$ alebo $\text{Rec}(k,g)$.

5.1.1 Sage grafy ako vstupné dátá

Program Sage ponúka zopár vopred naimplementovaných grafov, ktoré môžme využiť ako klietky $\text{Cage}(k,g)$. Na generovanie využijeme triedu `sage.graphs.graph_generators.GraphGenerators` [**sagemath**]). Trieda po- zostáva z konštruktorov pre niekoľko bežných grafov. Ku generátoru zástup- cov pristupujeme cez `graphs()` [**sagemath**]). Na zobrazenie známych gra- fov Sage využíva Spring-layout algoritmus. Z grafov využívame: **Peterse- nov graf** - `cage(3, 5)`, **Heawoodov graf** - `cage(3, 6)`, **McGeeho graf** - `cage(3, 7)`, **Tutteho-Coxeterov graf** - `cage(3, 8)`, **Balabanov graf** - `cage(3, 10)`, **Robertsonov graf** - `cage(4, 5)`, **Hoffmanov-Singletonov graf** - `cage(7, 5)`. Tieto grafy už majú vopred naimplementované všetky vrcholy a hrany medzi jednotlivými vrcholmi [**sagemath**].

5.1.2 Zoznam susedností ako vstupné dáta

Uvažujeme existujúce vstupné textové súbory, ktoré nesú informácie o susednostiach všetkých uvažovaných vrcholov jednotlivo pre 1 klietku $Cage(k,g)$ alebo rekordný graf $Rec(k,g)$. Riadok predstavuje zoznam vrcholov, s ktorými susedí konkrétny vrchol. Tieto vstupné súbory je možné získať na webovej stránke [Exo]. Navrhнемe si vlastný parser, ktorý nám rozdelí tieto dátu na zoznamy vrcholov. Z nich bude možné vytvoriť hrany. Tieto hrany bude možné pridať prázdnemu grafu (pomocou funkcie $EmptyGraph()$ [**sagemath**]) a tým z neho vytvoriť $Cage(k,g)$ alebo $Rec(k,g)$.

5.1.3 Vlastný návrh generovania

Generovanie klietok bez vstupných dát nemá jednotný prístup, je potrebné k nim pristupovať jednotlivo alebo preskúmať skupiny, ktoré sa generujú podobným spôsobom [EJ13].

Generovanie klietky Cage(6,4)

Navrhнемe vlastný spôsob generovania klietky $Cage(k,g)$ na základe parametrov k a g bez ďalších vstupných parametrov. Budúcu klietku vygenerujeme ako bipartitný graf. V Sage použijeme funkciu $DegreeSequenceBipartite(s,s)$ [**sagemath**], ktorá bude mať 2 rovnaké parametre s , pričom každý predstavuje zoznam vrcholov. Najskôr je potrebné si vypočítať Moorové ohraňčenie pre klietku $M(k,g)$ pre minimálny počet vrcholov. $Cage(6,4)$ je taký typ klietky, ktorej počet vrcholov $|ver|$ je rovnaký ako minimálny počet vrcholov z $M(k,g)$ [Mal07]. Tieto vrcholy rozdelíme na 2 zoznamy takým spôsobom,

že každý zoznam bude obsahovať $\frac{m}{2}$ vrcholov stupňa k a teda platí:

$$s = [k, k, k, k, k, k] \quad len(s) = \frac{m}{2} \quad (5.1)$$

Budúcej klietke $Cage(k,g)$ nastavíme vrcholy pomocou funkcie `set_vertex()` [sagemath]. Výsledný graf bude $cage(6,4)$

5.2 Validácia získaných klietok alebo rekordných grafov

Na validáciu existencie klietok $Cage(k,g)$ alebo rekordných grafov $Rec(k,g)$ bude potrebné zistieť Moorové ohraničenie pre klietku $M(k,g)$ na základe ktorého vieme otestovať počet vrcholov $|ver|$ ako aj existenciu klietky $Cage(k,g)$. Ako vstup využijeme parametre stupeň k a obvod g. Na základe vstupných parametrov vieme vypočítať Moorové ohraničenie pre klietku $M(k,g)$, ktoré porovnáme so skutočným počtom vrcholov $|ver|$ našej klietky $Cage(k,g)$. Následne vieme otestovať aj počet hrán $|edg|$, ktoré vieme rovnako na základe Moorovho ohraničenia $M(k,g)$ vypočítať.

5.3 Získavanie údajov z klietok alebo rekordných grafov

Sage ponúka zopár vopred naimplementovaných funkcií na získanie údajov z klietky $Cage(k,g)$ alebo rekordného grafu $Rec(k,g)$, niektoré však bude potrebné naprogramovať. Údaje ktoré budeme pri $Cage(k,g)$ alebo $Rec(k,g)$ sledovať budú

vrcholy a hrany , informačný pomer R ako aj **Počet automorfizmov**

získaný z klietky $|\text{AutGroup}(\text{Cage})|$ zistený priamo z $\text{Cage}(k,g)$ alebo $\text{Rec}(k,g)$. Špeciálny údaj bude **incidenčná matica**, ktorú budeme v ďalšej časti spracovávať v spojitosti s lineárnym kódom $C(n,m,d)$.

5.3.1 Počty vrcholov, hrán a automorfizmov, informačný pomer

Z vygenerovanej klietky $\text{Cage}(k,g)$ alebo rekordného grafu $\text{Rec}(k,g)$ v podobe grafu, ktorému sme na základe zoznamu susednosti zadefinovali vrcholy a prepojenia alebo na zaklade grafu, ktorý ich uz mal zadefinované vieme získať zoznam všetkých vrcholov pomocou funkcie $\text{vertices}()$ [**sagemath**]), zoznam všetkých hrán (pomocou funkcie $\text{edges}()$ [**sagemath**]) a zoznam všetkých automorfizmov(pomocou $\text{automorphism_group}()$ [**sagemath**]). Pre zistenie Počet vrcholov grafu $|ver|$, Počet hrán grafu $|edg|$ a Počet automorfizmov získaný z klietky $|\text{AutGroup}(\text{Cage})|$ nám stačí iba zistiť veľkosť ich zoznamov. Informačný pomer R udávajúci počet informácií alebo bitov nad celkovým počtom prenesených bitov vypočítame pomocou už známych počtov riadkov kontrolnej matice $|ver|$ a stĺpcov kontrolnej matice $|edg|$.

5.3.2 Incidenčné matice

Z klietky $\text{Cage}(k,g)$ alebo rekordného grafu $\text{Rec}(k,g)$ vieme získať incidenčnú maticu (pomocou funkcie $\text{incidence_matrix}()$ [**sagemath**]). Incidenčná matica predstavuje zároveň kontrolnú maticu lineárneho kódu H .

5.4 Lineárny kód a generujúca matica získaná z incidenčnej matice klietky

Z incidenčnej matice H je možné získať Lineárny kód $C(n,m,d)$ pomocou existujúcej funkcie v Sage `codes.from_parity_check_matrix(H)` [sagemath] a následne z neho Generujúcu maticu lineárneho kódu G pomocou funkcie `systematic_generator_matrix()`. Ďalej nás bude zaujímať **Minimálna Hammingova vzdialenosť** v kóde d , **Počet slov v kóde m**, **Grupa automorfizmov získaná z lineárneho kódu AutGroup(C)**, porovnanie s optimálnymi lineárnymi kódmi a s perfektnými lineárnymi kódmi

5.4.1 Minimálna kódová vzdialenosť, maximálny počet slov a grupa automorfizmov

Minimálnu vzdialenosť v kóde d je možné zistiť z Lineárneho kódu pomocou funkcie `minimum_distance()` [sagemath]

Počet slov v kóde m je možné zistiť z Generujúcej matice G nasledujúcim výpočtom:

$$m = 2^r \quad (5.2)$$

Parameter r bude predstavovať počet lineárne nezávislých riadkov generujúcej matice G

Grupu automorfizmov AutGroup(C) je možné zistiť z lineárneho kódu $C(n,m,d)$ pomocou funkcie `permutation_automorphism_group()` a **Počet automorfizmov získaný z lineárneho kódu |AutGroup(C)|** je možné zistiť z grupy automorfizmov $\text{AutGroup}(C)$ pomocou funkcie `order()` [sagemath]

5.4.2 Porovnanie s optimálnymi kódmi

Zadefinujeme si parameter $o(n,m,d)$ na základe vety č.2 o optimálnom kóde ako podiel nášho počtu slov a tabuľkovej hodnoty pre optimálny kód $A_2(n, d)$:

$$o(n, m, d) = \frac{m}{A_2(n, d)} \quad (5.3)$$

Tento parameter bude v intervale $<0,1>$ a bude znazorňovať akým spôsobom sa líšime od optimálneho kódu ($o(n,m,d) = 1$ by znamenalo, že uvažujeme optimálny kód).

5.4.3 Porovnanie s perfektnými kódmi

Zadefinujeme si parameter $p(n,m,d)$ na základe vety č.3 o perfektnom kóde ak vieme, že parameter $d = 2t + 1$:

$$p(n, m, d) = \frac{m \left\{ \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{t} \right\}}{2^n} \quad (5.4)$$

Tento parameter bude v intervale $<0,1>$ a bude znazorňovať akým spôsobom sa líšime od perfektného kódu ($p(n,m,d) = 1$ by znamenalo, že kód je perfektný).

5.5 Validácia lineárneho kódu

Na validáciu využijeme vzťah z definície lineárneho kódu $C(n,m,d)$ a jeho kontrolnej matice H :

$$G \times H^T = 0 \quad (5.5)$$

Vstupmi validácie sú kontrolná matica H a generujúca matica G . Súčin

generujúcej matice G a transponovanej kontrolnej matice H sa musí rovnať nulovej matice, ktorá má počet riadkov rovnajúci sa počtu riadkov generujúcej matice G a počet stĺpcov rovnajúci sa počtu stĺpcov transponovanej kontrolnej matice H

Kapitola 6

Riešenie

Riešenie sa zakladá na využití knižnice Sage [**sagemath**] dvoma spôsobmi. Prvé riešenie je implementované v online aplikácii CoCalc [Sag20] vo forme Sage skriptov. Druhé riešenie je implementované ako konzolová aplikácia v programovacom jazyku Python, kde bude knižnica Sage naimportovaná.

6.1 Generovanie klietky, rekordného grafu a generujúcej matice

V tejto kapitole uvádzame výsledky spracované v online aplikácii CoCalc a vo vlastnej konzolovej aplikácii klietky spustiteľnej na osobnom počítači v konzole programu Sage. Pre každú klietku $\text{Cage}(k,g)$ alebo rekordný graf $\text{Rec}(k,g)$ a lineárny kód $C(n,m,d)$ uvádzame tabuľku dosiahnutých výsledkov. Pri klietkach $\text{Cage}(k,g)$ nás zaujíma: počet vrcholov $|\text{ver}|$, počet hrán $|\text{ver}|$, rozmer kontrolnej matice H , počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ získaný z klietky $\text{Cage}(k,g)$, Moorovo ohraničenie $M(k,g)$ pre klietku a informačný pomer R . Pri lineárnych kódoch $C(n,m,d)$ nás zaujíma minimálna vzdialenosť d , počet slov v kóde vzdialenosť m , rozmer generujúcej matice vzdialenosť

G , počet automorfizmov $|\text{AutGroup}(C)|$ zistený z lineárneho kódu, parameter perfektného kódu $p(n,m,d)$ a parameter optimálneho kódu $o(n,m,d)$ (ak je možné získať). Tieto dva parametre je možné následne medzi sebou porovnať pri vyhodnocovaní získaného lineárneho kódu $C(n,m,d)$. Pri vytváraní textových súborov s kontrolnými maticami H a generujúcimi maticami G budeme sledovať aj ich dobu vytvárenia a uloženia $t(H)$ $t(G)$. Pri sledovaní týchto časov budeme vychádzať z prvého riešenia, nakoľko v konzolovej aplikácii sú časy ovplyvnené tým, že je aplikácia spustená. Počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ zistených z klietky porovnávame s počtom automorfizmov $|\text{AutGroup}(C)|$ získaných z lineárneho kódu $C(n,m,d)$. Uvádzame výsledky všetkých typov uvažovaných klietok $\text{Cage}(k,g)$ a rekordných grafov $\text{Rec}(k,g)$ od ich vygenerovania až po získanie lineárneho kódu $C(n,m,d)$ spolu s generujúcou maticou G . Pri niektorých výsledkoch zobrazujeme aj kontrolnú H , generujúcu maticu G a graf klietky $\text{Cage}(k,g)$, pri zložitejších grafoch je už zobrazenie obtiažne, preto ich neuvádzame avšak matice sú dostupné v textových súboroch.

6.1.1 Dôležité funkcie

V tejto podkapitole si uvedieme niektoré funkcie, ktoré bolo potrebné naprogramovať.

Výpočet Moorovho ohraničenia $M(k,g)$

```
# vypocita najmensi mozny pocet vrcholov pre potencionalnu klietku (testovacie data)
def calculateMooreBound(self):
    M = 0
    exp = 0
    if self.g % 2 == 1:
        exp = (self.g - 3) / 2
        for i in range(int(exp) + 1):
            M = M + (self.k * ((self.k - 1) ** i))
        M = M + 1
    else:
        exp = (self.g - 2)/2
        for i in range(int(exp) + 1):
            M = M + ((self.k - 1) ** i)
        M = 2 * M
    return M
```

Obr. 6.1: $M(k,g)$ - riešenie

Konštrukcia klietok zo vstupných dát

```
# vkladanie do súboru uložených klietok v ktorom potencionálne je možné, aby boli vložené aj pravidlá generovania klietiek
def get Cage(self):
    sageCages = [[], [2,2], [2,3], [2,7], [2,8], [2,10], [2,11], [2,5], [2,3]]
    exosAdjacencyList = [[], [1,2,3], [1,3,4], [1,2,4], [1,2,3], [1,3,5], [1,4,7], [4,7], [4,10], [5,10], [7,11], [7,8], [10,5], [11,4], [12,5], [13,5]]
    myConstructedCages = [[], [5,4]]
    cageInput = []
    cageInput.append(str(self.k))
    cageInput.append(str(self.g))

    if ((cageInput in sageCages) == False and (cageInput in exosAdjacencyList) == False and (cageInput in myConstructedCages) == False):
        print("!")
        print("ODE_MOR_EXIST")
        return

    cage = graphs.EmptyGraph()
    if self.g <= 2:
        cage = graphs.CageGraph()
    if self.g > 2 and self.g <= 3:
        if self.k == 3:
            if self.g == 3:
                cage = graphs.PetersenGraph()
            if self.g == 6:
                cage = graphs.HeawoodGraph()
            if self.g == 7:
                cage = graphs.McGeeGraph()
            if self.g == 8:
                cage = graphs.TutteCoxeterGraph()
            if self.g == 10:
                cage = graphs.BalancedCage()
            if self.g == 11:
                cage = graphs.Balanced1Cage()
        if self.k == 4 and self.g == 5:
            cage = graphs.DodecahedronGraph()
        if self.k == 7 and self.g == 5:
            cage = graphs.HoffmanSingletonGraph()
    elif cageInput in myConstructedCages:
        cage = self.generateGraphFromExosAdjacencyList(self.const.EXOS_CAGE_PREFIX + str(self.k) + "-" + str(self.g) + self.const.TEXT_FILE)
    elif cageInput in sageCages:
        cage = self.generateSageGraph()
    if cage != None:
        cage = self.generate_6_4cagedGraph()
    if (self.cageVerification(cage) == False):
        print("ODE_MOR")
    else:
        print("ODE_MOR_EXIST")
    cageInput = []
    return cage
```

Obr. 6.2: Konštrukcia klietky Cage(k,g) - riešenie

Generovanie klietky Cage(6,4)

```
# vygeneruje Cage(6,4), bez vstupnych parametrov ako bipartitny graf - individualny pristup
def generate_6_4CageGraph(self):
    self.k = 6
    self.g = 4
    M = self.calculateMooreBound()
    s = []
    for i in range(int(M/2)):
        s.append(6)
    graph = graphs.DegreeSequenceBipartite(s,s)
    for i in range (M):
        graph.set_vertex(i, str(i))
    graph.name(self.const.CAGE_64)
    return graph
```

Obr. 6.3: Konštrukcia Cage(6,4) - riešenie

Parser na spracovanie textových súborov zoznamov susednosti

```
# odstranenie hodnot zo zoznamu na zaklade hodnoty
def remove_values_from_list(self,the_list, val):
    return [value for value in the_list if value != val]

# parsovanie Exoo dokumentov so zoznamami susednych vrcholov na generovanie grafu
def generateGraphandCageFromExoosAdjacencylist(self,filePath):
    file = open(filePath, "r")
    graph = graphs.EmptyGraph()
    mainVertex = 0
    for line in file:
        line = line.replace('\n', '')
        adjacencyVerticies = line.split(" ")
        adjacencyVerticies = self.remove_values_from_list(adjacencyVerticies, '')
        for adjacencyVertex in adjacencyVerticies:
            graph.add_edge(mainVertex, int(adjacencyVertex))
        mainVertex += 1
    file.close()
    return graph
```

Obr. 6.4: Parser zoznamu susednosti - riešenie

Validácia klietok Cage(k,g)

```
# vypocita potencionalny pocet hran na zaklade Moorovho ohranicenia (testovacie data)
def number_of_edges_by_moore_bound(self, M):
    return M * self.k / 2

# test vysledkov
def cage_verification(self, cage):
    M = self.calculate_moore_bound()
    min_pocet_hran = self.number_of_edges_by_moore_bound(M)
    # verifikacia poctu vrcholov
    if len(cage.vertices()) < M or len(cage.incidence_matrix().column(0)) < M:
        return False
    # verifikacia poctu hran
    if len(cage.edges()) < min_pocet_hran or len(cage.incidence_matrix().row(0)) < min_pocet_hran:
        return False
    return True
```

Obr. 6.5: Validácia klietok - riešenie

Výpočet parametra perfektného kódu p(n,m,d)

```
# udava vzdialenosť nášho kódu od perfektného v intervale 0 - 1, kedy 1 je perfektný kód
def get_perfect_code_parameter(self, n, M, d):
    if d % 2 == 1:
        t = (d-1)/2      # t=(d-1)/2 pre nepárne
    else:
        t = (d-2)/2      # parné (d-1)/2 dostavam desatinne číslo, uvažujem len cele, preto d-2
    sum = 0.0
    for i in range(0, int(t) + 1):
        sum = sum + self.combination_number(n, i)
    parameter = (M * sum) / (2 ** n)
    return parameter

# vypočet faktoriálu
def factorial(self, n):
    fact = 1
    for i in range(1, n+1):
        fact = fact * i
    return fact

# vypočet kombináčneho čísla
def combination_number(self, n, k):
    return self.factorial(n) / (self.factorial(k) * self.factorial(n-k))
```

Obr. 6.6: Parameter perfektného kódu - riešenie

Spracovanie matíc H, G

Ukladanie H do textových súborov

```
# vygeneruje Kontrolnu maticu, ulozi ju a sleduje cas
def main():
    cages = getSortedCagesByMooreBound();
    for cage in cages:
        k = cage[0];
        g = cage[1];
        startTime = datetime.datetime.now();
        H = getParityCheckMatrixFromCage(k,g);
        file = open('ParityCheckMatrices/H_cage_'+str(k)+'_'+str(g)+'.txt', 'w');
        file.write(str(H));
        time = datetime.datetime.now() - startTime;
        print('Súbor H_Cage('+str(k)+','+str(g)+') bol úspešne vytvorený! Doba trvania: '+str(time.total_seconds())+'s\n');
        file.close();

# zoradi kletky podla Moorovo ohranicenia
def getSortedCagesByMooreBound():
    sageCages = myConstructedCages + exoosCages;
    sortedCages = [];
    for cage in sageCages:
        k = cage[0];
        g = cage[1];
        cage.append(calculateMooreBound(k,g))
    sortedCages.append(cage);
    sortedCages.sort(key=takeThird);
    for i in range(len(sortedCages)):
        del sortedCages[i][2];
    return sortedCages;
```

Obr. 6.7: Uloženie matice H do textoveho suboru - riešenie

Spracovanie H a ukladanie G do textových súborov

```
# zoradi textove subory kontrolnych matic podla velkosti, nacita ich a zobrazí data, vygeneruje generujucu maticu, ulozi ju a sleduje cas
def main():
    filepaths = glob.glob('ParityCheckMatrices/*.txt');
    sortedfilepaths = sortParityCheckFilesBySize(filepaths);
    for i in range(len(sortedfilepaths) - 1):
        startTime = datetime.datetime.now();
        if os.path.getsize(sortedfilepaths[i]) > 0:
            file = open(sortedfilepaths[i], 'r');
            H = getParityCheckMatrixFromFile(file);
            file.close();
            G = getGeneratorMatrix(H);
            sortedfilepaths[i] = sortedfilepaths[i].replace('ParityCheckMatrices', 'GeneratorMatrices');
            sortedfilepaths[i] = sortedfilepaths[i].replace('H', 'G');
            file = open(sortedfilepaths[i], 'w');
            file.write(str(G));
            sortedfilepaths[i] = sortedfilepaths[i].replace('GeneratorMatrices','');
            time = datetime.datetime.now() - startTime;
            print('Súbor ' + str(sortedfilepaths[i]) + ' bol úspešne vytvorený! Doba vytvárania súboru: '+str(time.total_seconds())+'s\n');
            file.close();
        else:
            print('Súbor ' + str(sortedfilepaths[i]) + ' je prázdný! Nemôžem vytvoriť generujuúcu maticu!\n');
            continue;
    # zoradi textove subory na zaklade velkosti
    def sortParityCheckFilesBySize(filepaths):
        filepaths = sorted(filepaths, key = lambda x: os.stat(x).st_size);
    return filepaths;
```

Obr. 6.8: Získanie matice H a uloženie matice G do textového súboru - riešenie

Validácia lineárneho kódu C(n,m,d)

```

# ak je sumu vsetkych elementov maticu G a Ht rovny 0 a sucasne platí ze ma tato matica
# rovnaky pocet riadkov ako G a rovnaky pocet stlpcov ako Ht potom je kod validny
def isLinearCodeValid(self,G,H):
    Ht = H.transpose()
    validationMatrix = G * Ht
    elementsSum = sum(sum(validationMatrix))
    if len(G.column(0)) == len(validationMatrix.column(0)) and len(Ht.row(0)) == len(validationMatrix.row(0)) and elementsSum == 0:
        return True
    else:
        return False

```

Obr. 6.9: Validácia lineárneho kódu - riešenie

Získanie počtu automorfizmov

Z klietky Cage(k,g):

```

cage = graph.getCage()
autGroup = len(cage.automorphism_group())

```

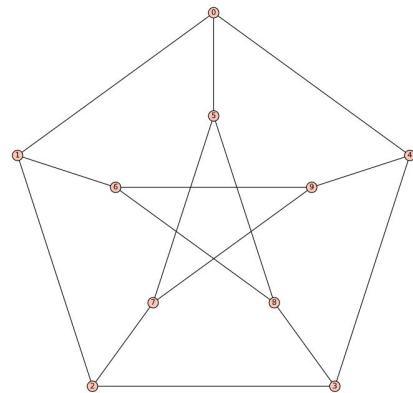
Obr. 6.10: Počet automorfizmov z klietky - riešenie

Z lineárneho kódu $C(n,m,d)$:

```
# zisti pocet automorfizmov z kodu
def getNumberOfAutomorphism(self,H):
    C = self.getLinearCode(H)
    Gr = C.permutation_automorphism_group(algorithm = "partition")
    return Gr.order()
```

Obr. 6.11: Počet automorfizmov z lineárneho kódu - riešenie

6.1.2 Cage(3,5) - Petersenov graf



Obr. 6.12: Cage(3,5) [sagemath]

[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]	
[1 0 0 1 1 0 0 0 0 0 0 0 0 0 0]	
[0 0 0 1 0 1 1 0 0 0 0 0 0 0 0]	
[0 0 0 0 1 0 1 1 0 0 0 0 0 0 0]	
[0 1 0 0 0 0 1 0 1 0 0 0 0 0 0]	
[0 0 1 0 0 0 0 0 0 1 1 0 0 0]	
[0 0 0 0 1 0 0 0 0 0 0 1 1 0]	
[0 0 0 0 0 1 0 0 0 1 0 0 0 1]	
[0 0 0 0 0 0 0 1 0 0 1 1 0 0]	
[0 0 0 0 0 0 0 1 0 0 0 1 1 1]	

[1 0 1 0 1 0 0 0 0 0 0 1 1 0 0]	
[0 1 1 0 0 0 0 0 0 1 0 1 1 1 0]	
[0 0 0 1 1 0 1 0 0 0 0 0 0 1 1]	
[0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1]	
[0 0 0 0 0 0 1 1 1 0 0 0 1 1 1]	
[0 0 0 0 0 0 0 1 1 1 0 0 1 1 0]	
[0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	

Obr. 6.13: H_Cage(3,5) (vľavo) a G_Cage(3,5) (vpravo)

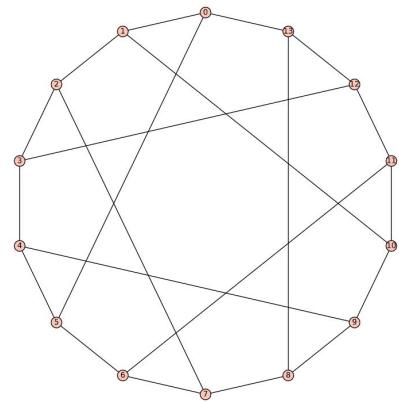
Cage(3,5)		C(15,6,5)	
ver	10	d	5
edg	15	m	64
H	10×15	G	6×15
t(H)	0,417s	t(G)	0,005s
AutGroup(Cage)	120	AutGroup(C)	120
M(k,g)	10	p(n,m,d)	0,236328
R	1/3	o(n,m,d)	0,25

Tabuľka 6.1: Tabuľka Cage(3,5)

V tabuľke Cage(3,5) môžme vidieť, že klietka má rovnaký počet vrcholov |ver| ako Moorove ohraničenie M(3,5). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,417s. Takisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice klietky H sme vedeli určiť lineárny kód C(15,6,5), ktorý má počet kódových slov m=64. Rovnako vieme určiť aj rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,005s. Takisto bolo možné určiť z lineárneho kódu C(15,6,5) počet automorfizmov

$|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$ a optimálneho kódu $o(n,m,d)$, ktoré sme medzi sebou porovnali a zistili mierny rozdiel.

6.1.3 Cage(3,6) - Heawoodov graf



Obr. 6.14: Cage(3,6) [sagemath]

```
[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0]
```

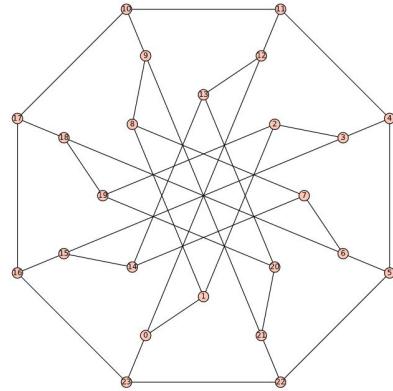
Obr. 6.15: H_Cage(3,6) (vľavo) a G_Cage(3,6) (vpravo)

Cage(3,6)		C(21,8,6)	
ver	14	d	6
edg	21	m	256
H	14×21	G	8×21
t(H)	0,007s	t(G)	0,007s
AutGroup(Cage)	336	AutGroup(C)	336
M(k,g)	14	p(n,m,d)	0,028320
R	1/3	o(n,m,d)	0,1

Tabuľka 6.2: Tabuľka Cage(3,6)

V tabuľke Cage(3,6) môžme vidieť, že klietka má rovnaký počet vrcholov |ver| ako Moorove ohraničenie M(3,6). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,007s. Takisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice klietky H sme vedeli určiť lineárny kód C(21,8,6), ktorý má počet kódových slov m=256. Rovnako vieme určiť aj rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,007s. Takisto bolo možné určiť z lineárneho kódu C(21,8,6) počet automorfizmov |AutGroup(C)|, ktorý je zhodný s počtom automorfizmov z klietky |AutGroup(Cage)|. Určili sme aj parameter perfektného p(n,m,d) a optimálneho kódu o(n,m,d), ktoré sme medzi sebou porovnali a zistili mierny rozdiel.

6.1.4 Cage(3,7) - McGeeho graf



Obr. 6.16: Cage(3,7) [sagemath]

Obr. 6.17: H₋Cage(3,7)

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Obr. 6.18: G_Cage(3,7)

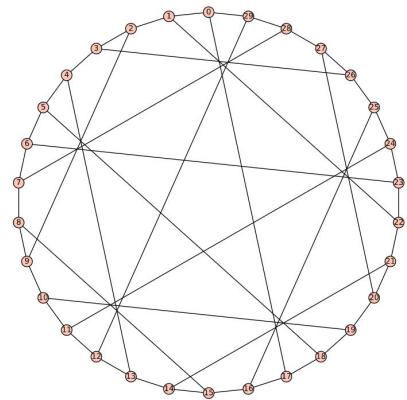
Cage(3,7)		C(36,13,7)	
ver	24	d	7
edg	36	m	8192
H	24×6	G	13×36
t(H)	0,006s	t(G)	0,012s
AutGroup(Cage)	32	AutGroup(C)	32
M(k,g)	22	p(n,m,d)	0,000931
R	1/3	o(n,m,d)	-

Tabuľka 6.3: Tabuľka Cage(3,7)

V tabuľke Cage(3,7) môžme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(3,7). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,006s. Takisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice klietky H sme vedeli určiť lineárny kód C(36,13,7), ktorý má počet kódových slov m=8192. Rovnako vieme určiť aj rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,012s. Takisto bolo možné určiť z lineárneho kódu C(36,13,7) po-

čet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$, ktorý ale nemôžme porovnať s parametrom optimálneho kódu $o(n,m,d)$, pretože pre dĺžku kódu $n=36$ tento údaj nie je známy.

6.1.5 Cage(3,8) - Tutteho-Coxeterov graf



Obr. 6.19: Cage(3,8) [sagemath]

Obr. 6.20: H-Cage(3,8)

Obr. 6.21: G_Cage(3,8)

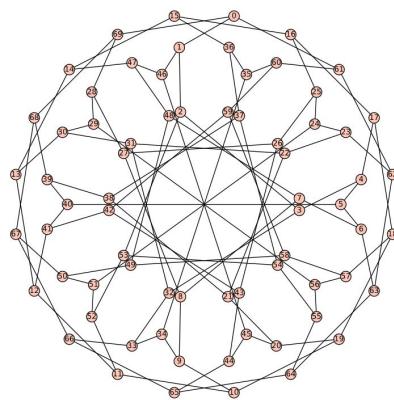
Cage(3,8)		C(45,16,8)	
ver	30	d	8
edg	45	m	65536
H	30×45	G	16×45
t(H)	0,009s	t(G)	0,021s
AutGroup(Cage)	1440	AutGroup(C)	1440
M(k,g)	30	p(n,m,d)	0,000028
R	1/3	o(n,m,d)	-

Tabul'ka 6.4: Tabul'ka Cage(3,8)

V tabuľke Cage(3,8) môžme vidieť, že klietka má rovnaký počet vrcholov $|ver|$ ako Moorove ohraničenie $M(3,8)$. Na základe počtu vrcholov $|ver|$ a hrán $|edg|$ bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,009s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R . Z incidenčnej matice klietky H sme vedeli určiť lineárny kód $C(45,16,8)$, ktorý má počet kódových slov $m=65536$. Rovnako vieme určiť aj rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,021s. Takisto bolo možné určiť z lineárneho kódu $C(45,16,8)$ počet automor-

fizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$, ktorý ale nemôžme porovnať s parametrom optimálneho kódu $o(n,m,d)$, pretože pre dĺžku kódu $n=45$ tento údaj nie je známy.

6.1.6 Cage(3,10) - Balabanov graf



Obr. 6.22: Cage(3,10) [sagemath]

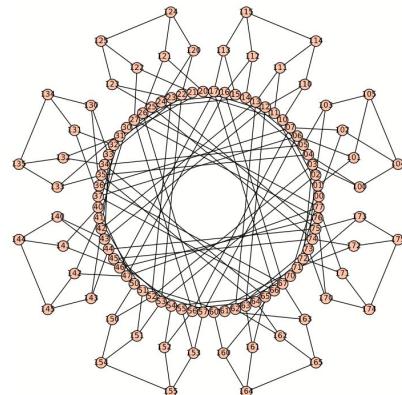
Cage(3,10)		C(105,36,?)	
$ \text{ver} $	70	\mathbf{d}	-
$ \text{edg} $	105	\mathbf{m}	68719×10^6
\mathbf{H}	70×105	\mathbf{G}	36×105
$t(H)$	0,039s	$t(G)$	0,091s
$ \text{AutGroup}(\text{Cage}) $	80	$ \text{AutGroup}(C) $	80
$M(k,g)$	62	$p(n,m,d)$	-
R	$1/3$	$o(n,m,d)$	-

Tabuľka 6.5: Tabuľka Cage(3,10)

V tabuľke Cage(3,10) môžme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov $|\text{ver}|$ ako Moorove ohraničenie $M(3,10)$. Na základe počtu vrcholov

$|ver|$ a hrán $|edg|$ bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,039s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R . Z incidenčnej matice klietky H sme nevedeli určiť úplný lineárny kód, iba $C(45,16,?)$, ktorý má počet kódových slov $m=68719 \times 10^6$. JE to z toho dôvodu, že pre program je už príliš náročné vypočítať minimálnu vzdialenosť v kóde d . Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,091s. Takisto bolo možné určiť z lineárneho kódu $C(45,16,?)$ počet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Parameter perfektného $p(n,m,d)$ a parametrer optimálneho kódu $o(n,m,d)$ nie je možné určiť, pretože nepoznáme d .

6.1.7 Cage(3,11) - Balabanov graf



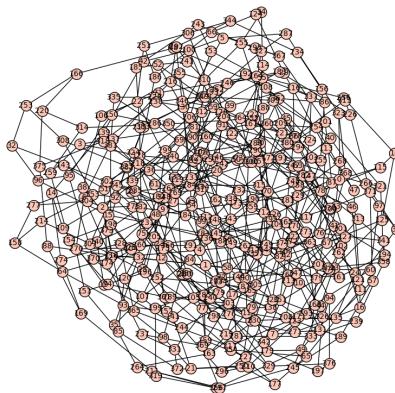
Obr. 6.23: Cage(3,11) [sagemath]

Cage(3,11)		C(168,57,?)	
ver	112	d	-
edg	168	m	14412×10^{13}
H	112×168	G	57×168
t(H)	0,095s	t(G)	0,22s
AutGroup(Cage)	64	AutGroup(C)	64
M(k,g)	94	p(n,m,d)	-
R	1/3	o(n,m,d)	-

Tabuľka 6.6: Tabuľka Cage(3,11)

V tabuľke Cage(3,11) môžme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(3,11). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,095s. Takisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice klietky H sme nevedeli určiť úplný lineárny kód, iba C(168,57,?), ktorý má počet kódových slov $m=14412 \times 10^{13}$. Je to z toho dôvodu, že pre program je už príliž náročné vypočítať minimálnu vzdialenosť v kóde d. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,22s. Takisto bolo možné určiť z lineárneho kódu C(168,57,?) počet automorfizmov |AutGroup(C)|, ktorý je zhodný s počtom automorfizmov z klietky |AutGroup(Cage)|. Parameter perfektného p(n,m,d) a parameter optimálneho kódu o(n,m,d) nie je možné určiť, pretože nepoznáme d.

6.1.8 Rec(3,14)



Obr. 6.24: Rec(3,14)

Rec(3,14)		C(576,193,?)	
ver	384	d	-
edg	576	m	12554×10^{54}
H	384×586	G	193×576
t(H)	1,26s	t(G)	2,627s
AutGroup(Cage)	96	AutGroup(C)	96
M(k,g)	254	p(n,m,d)	-
R	1/3	o(n,m,d)	-

Tabuľka 6.7: Tabuľka Rec(3,14)

V tabuľke Rec(3,14) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(3,14). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 1,26s. Ta- kisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice rekordného grafu H sme neve- deli určiť úplný lineárny kód, iba C(576,193,?), ktorý má počet kódových

slov $m=12554 \times 10^{54}$. Je to z toho dôvodu, že pre program je už príliž náročné vypočítať minimálnu vzdialenosť v kóde d. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 2,627s. Takisto bolo možné určiť z lineárneho kódu $C(576,193,?)$ počet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Parameter perfektného $p(n,m,d)$ a parametrer optimálneho kódu $o(n,m,d)$ nie je možné určiť, pretože nepoznáme d.

6.1.9 Rec(3,16)

Rec(3,16)		C(1440,481,?)	
ver	960	d	-
edg	1440	m	62435×10^{140}
H	960×1440	G	481×1440
t(H)	7,389s	t(G)	17,603s
AutGroup(Cage)	96	AutGroup(C)	96
M(k,g)	510	p(n,m,d)	-
R	1/3	o(n,m,d)	-

Tabuľka 6.8: Tabuľka Rec(3,16)

V tabuľke Rec(3,16) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(3,16). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 7,389s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R. Z incidenčnej matice rekordného grafu H sme nevedeli určiť úplný lineárny kód, iba C(1440,481,?), ktorý má počet kódových slov $m=62435 \times 10^{140}$. Je to z toho dôvodu, že pre program je už príliž náročné vypočítať minimálnu vzdialenosť v kóde d. Vieme určiť rozmer generujú-

cej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 17,603s. Takisto bolo možné určiť z lineárneho kódu C(1440,481,?) počet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Parameter perfektného p(n,m,d) a parametrer optimálneho kódu o(n,m,d) nie je možné určiť, pretože nepoznáme d.

6.1.10 Rec(3,17)

Rec(3,17)		C(3264,1089,?)	
ver	2176	d	-
edg	3264	m	66323×10^{323}
H	2176×3264	G	1089×3264
t(H)	37,479s	t(G)	87,334s
AutGroup(Cage)	544	AutGroup(C)	544
M(k,g)	766	p(n,m,d)	-
R	1/3	o(n,m,d)	-

Tabuľka 6.9: Tabuľka Rec(3,17)

V tabuľke Rec(3,17) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(3,17). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 37,479s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R. Z incidenčnej matice rekordného grafu H sme nevedeli určiť úplný lineárny kód, iba C(3264,1089,?), ktorý má počet kódových slov $m=66323 \times 10^{323}$. Je to z toho dôvodu, že pre program je už príliž náročné vypočítať minimálnu vzdialenosť v kóde d. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 87,334s. Takisto bolo možné určiť z lineárneho kódu C(3264,1089,?) počet automor-

fizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Parameter perfektného $p(n,m,d)$ a parametrer optimálneho kódu $o(n,m,d)$ nie je možné určiť, pretože nepoznáme d.

6.1.11 Rec(3,18)

Rec(3,18)		C(3840,1281,?)	
ver	2560	d	-
edg	3840	m	41632×10^{381}
H	2560×3840	G	1281×3840
t(H)	57,305s	t(G)	121,156s
AutGroup(Cage)	640	AutGroup(C)	640
M(k,g)	1022	p(n,m,d)	-
R	1/3	o(n,m,d)	-

Tabuľka 6.10: Tabuľka Rec(3,20)

V tabuľke Rec(3,18) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(3,18). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 57,305s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R. Z incidenčnej matice rekordného grafu H sme nevedeli určiť úplný lineárny kód, iba C(3840,1281,?), ktorý má počet kódových slov $m=41632 \times 10^{381}$. Je to z toho dôvodu, že pre program je už príliž náročné vypočítať minimálnu vzdialenosť v kóde d. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 121,156. Takisto bolo možné určiť z lineárneho kódu C(3840,1281,?) počet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Parameter perfektného $p(n,m,d)$ a parametrer optimál-

neho kódu $o(n,m,d)$ nie je možné určiť, pretože nepoznáme d.

6.1.12 Rec(3,20)

Rec(3,20)		C(?, ?, ?)	
ver	5376	d	-
edg	8064	m	-
H	5376×8064	G	-
t(H)	240,707s	t(G)	-
AutGroup(Cage)	2688	AutGroup(C)	-
M(k,g)	2046	p(n,m,d)	-
R	1/3	o(n,m,d)	-

Tabuľka 6.11: Tabuľka Rec(3,20)

V tabuľke Rec(3,20) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(3,20). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 240,707s. Takisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice rekordného grafu H sme už nevedeli určiť lineárny kód nakoľko sa jedná o výpočtovo príliž náročný problém.

6.1.13 Rec(3,23)

Rec(3,23)		C(?, ?, ?)	
ver	49326	d	-
edg	73989	m	-
H	49326×73987	G	-
t(H)	-	t(G)	-
AutGroup(Cage)	1	AutGroup(C)	-
M(k,g)	6142	p(n,m,d)	-
R	1/3	o(n,m,d)	-

Tabuľka 6.12: Tabuľka Rec(3,23)

V tabuľke Rec(3,23) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(3,23). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H. Samotnú maticu sa nám vytvoriť a uložiť už nepodarilo. Bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice rekordného grafu H sme už nevedeli určiť lineárny kód nakoľko sa jedná o výpočtovo príliž náročný problém.

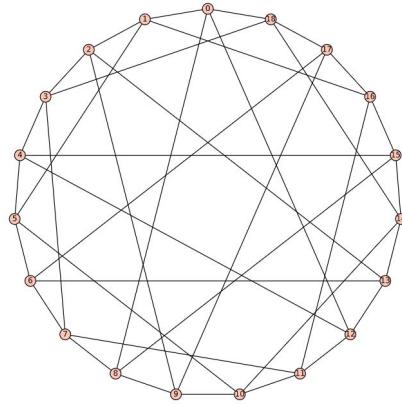
6.1.14 Rec(3,25)

Rec(3,25)		C(?, ?, ?)	
ver	108906	d	-
edg	163359	m	-
H	108906 163359	G	-
t(H)	-	t(G)	-
AutGroup(Cage)	-	AutGroup(C)	-
M(k,g)	12286	p(n,m,d)	-
R	1/3	o(n,m,d)	-

Tabuľka 6.13: Tabuľka Rec(3,25)

V tabuľke Rec(3,25) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(3,25). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a informačný pomer R. Samotnú maticu sa nám vytvoriť a uložiť už nepodarilo, pretože ide o výpočtovo veľmi náročný problém. Nebolo možné určiť z klietky ani počet automorfizmov |AutGroup(Cage)|.

6.1.15 Cage(4,5) - Robertsonov graf



Obr. 6.25: Cage(4,5) [sagemath]

```
[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
```

Obr. 6.26: H_Cage(4,5)

Obr. 6.27: G_Cage(4,5)

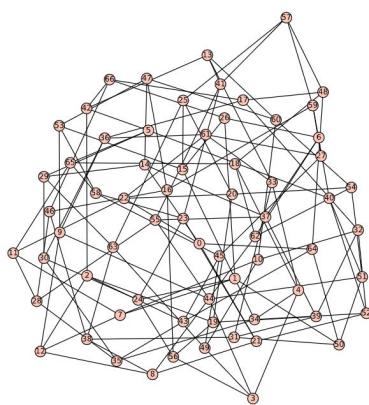
Cage(4,5)		C(38,20,5)	
ver	19	d	5
edg	38	m	1048576
H	19×38	G	20×38
t(H)	1,385s	t(G)	0,015s
AutGroup(Cage)	24	AutGroup(C)	24
M(k,g)	17	p(n,m,d)	0,002831
R	1/2	o(n,m,d)	-

Tabuľka 6.14: Tabuľka Cage(4,5)

V tabuľke Cage(4,5) môžme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov $|ver|$ ako Moorove ohraničenie $M(4,5)$. Na základe počtu vrcholov $|ver|$ a hrán $|edg|$ bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 1,385s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R . Z incidenčnej matice klietky H sme vedeli určiť lineárny kód $C(38,20,5)$, ktorý má počet kódových slov $m=1048576$. Rovnako vieme určiť aj rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,015s. Takisto bolo možné určiť z lineárneho kódu $C(38,20,5)$ po-

čet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$, ktorý ale nemôžme porovnať s parametrom optimálneho kódu $o(n,m,d)$, pretože pre dĺžku kódu $n=38$ tento údaj nie je známy.

6.1.16 Cage(4,7)



Obr. 6.28: Cage(4,7)

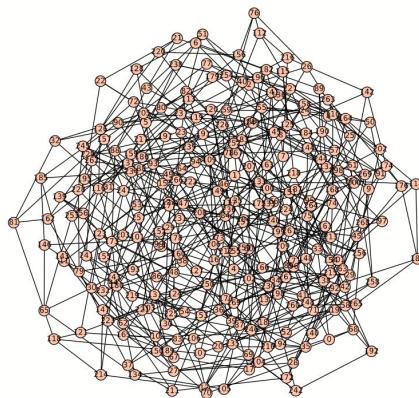
Cage(4,7)		C(134,68,7)	
ver	67	d	7
edg	134	m	29515×10^{16}
H	67×134	G	68×134
t(H)	0,046s	t(G)	0,144s
\text{AutGroup}(\text{Cage})	4	\text{AutGroup}(C)	4
M(k,g)	53	p(n,m,d)	$5,436 \times 10^{-15}$
R	1/2	o(n,m,d)	-

Tabuľka 6.15: Tabuľka Cage(4,7)

V tabuľke Cage(4,7) môžeme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(4,7). Na základe počtu vrcholov

$|ver|$ a hrán $|edg|$ bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,046s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R . Z incidenčnej matice klietky H sme vedeli určiť lineárny kód $C(134,68,7)$, ktorý má počet kódových slov $m=29515 \times 10^{16}$. Rovnako vieme určiť aj rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,144s. Takisto bolo možné určiť z lineárneho kódu $C(134,68,7)$ počet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$, ktorý ale nemôžme porovnať s parametrom optimálneho kódu $o(n,m,d)$, pretože pre dĺžku kódu $n=134$ tento údaj nie je známy.

6.1.17 Cage(4,9)



Obr. 6.29: Cage(4,9)

Cage(4,9)		C(540,271,?)	
ver	270	d	-
edg	540	m	37943×10^{77}
H	270×540	G	271×540
t(H)	0,827s	t(G)	2,257s
AutGroup(Cage)	90	AutGroup(C)	90
M(k,g)	161	p(n,m,d)	-
R	1/2	o(n,m,d)	-

Tabuľka 6.16: Tabuľka Cage(4,9)

V tabuľke Cage(4,9) môžme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(4,9). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,827s. Takisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice klietky H sme nevedeli určiť úplný lineárny kód, iba C(540,271,?), ktorý má počet kódových slov $m=37943 \times 10^{77}$. Je to z toho dôvodu, že pre program je už príliž náročné vypočítať minimálnu vzdialenosť v kóde d. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 2,257s. Takisto bolo možné určiť z lineárneho kódu C(540,271,?) počet automorfizmov |AutGroup(C)|, ktorý je zhodný s počtom automorfizmov z klietky |AutGroup(Cage)|. Parameter perfektného p(n,m,d) a parameter optimálneho kódu o(n,m,d) nie je možné určiť, pretože nepoznáme d.

6.1.18 Cage(4,10)

Cage(4,10)		C(768,385,?)	
ver	384	d	-
edg	768	m	78804×10^{111}
H	384×768	G	385×768
t(H)	1,535s	t(G)	4,707s
AutGroup(Cage)	768	AutGroup(C)	768
M(k,g)	242	p(n,m,d)	-
R	1/2	o(n,m,d)	-

Tabuľka 6.17: Tabuľka Cage(4,10)

V tabuľke Cage(4,10) môžme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(4,10). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 1,535s. Takisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice klietky H sme nevedeli určiť úplný lineárny kód, iba C(768,385,?), ktorý má počet kódových slov $m=78804 \times 10^{111}$. Je to z toho dôvodu, že pre program je už príliž náročné vypočítať minimálnu vzdialenosť v kóde d. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 4,707s. Takisto bolo možné určiť z lineárneho kódu C(768,385,?) počet automorfizmov |AutGroup(C)|, ktorý je zhodný s počtom automorfizmov z klietky |AutGroup(Cage)|. Parameter perfektného p(n,m,d) a parameter optimálneho kódu o(n,m,d) nie je možné určiť, pretože nepoznáme d.

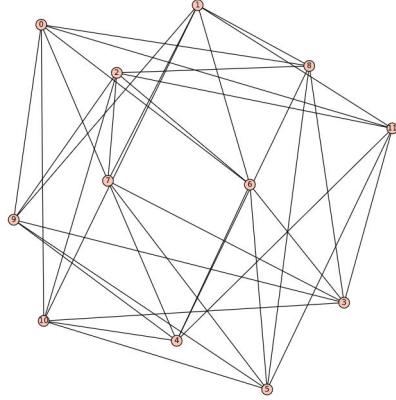
6.1.19 Cage(5,10)

Cage(5,10)		C(3240,1945,?)	
ver	1296	d	-
edg	3240	m	31867×10^{581}
H	1296×3240	G	1945×3240
t(H)	22,806s	t(G)	83,87s
AutGroup(Cage)	3888	AutGroup(C)	3888
M(k,g)	682	p(n,m,d)	-
R	3/5	o(n,m,d)	-

Tabuľka 6.18: Tabuľka Cage(5,10)

V tabuľke Cage(5,10) môžme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(5,10). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 22,806s. Takisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice klietky H sme nevedeli určiť úplný lineárny kód, iba C(3240,1945,?), ktorý má počet kódových slov $m=31867 \times 10^{581}$. Je to z toho dôvodu, že pre program je už príliž náročné vypočítať minimálnu vzdialenosť v kóde d. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 83,87s. Takisto bolo možné určiť z lineárneho kódu C(3240,1945,?) počet automorfizmov |AutGroup(C)|, ktorý je zhodný s počtom automorfizmov z klietky |AutGroup(Cage)|. Parameter perfektného p(n,m,d) a parameter optimálneho kódu o(n,m,d) nie je možné určiť, pretože nepoznáme d.

6.1.20 Cage(6,4)



Obr. 6.30: Cage(6,4)

Obr. 6.31: H₋Cage(6,4)

```
[1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1]
[0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1]
[0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1]
[0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1]
[0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1]
[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1]
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1]
[0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1]
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
[0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
```

Obr. 6.32: G_Cage(6,4)

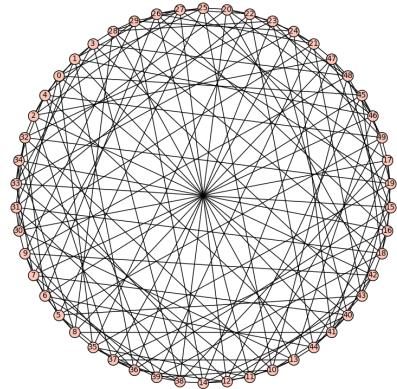
Cage(6,4)		C(36,25,4)	
ver	12	d	4
edg	36	m	33554432
H	12×36	G	25×36
t(H)	0,052s	t(G)	0,013s
AutGroup(Cage)	1036800	AutGroup(C)	1036800
M(k,g)	12	p(n,m,d)	0,018066
R	2/3	o(n,m,d)	-

Tabuľka 6.19: Tabuľka Cage(6,4)

V tabuľke Cage(6,4) môžme vidieť, že klietka má rovnaký počet vrcholov |ver| ako Moorove ohraničenie M(6,4). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,052s. Takisto bolo možné určiť z klietky počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice klietky H sme vedeli určiť úplný lineárny kód C(36,25,4), ktorý má počet kódových slov m=33554432. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,013s. Takisto bolo možné určiť z lineárneho kódu C(36,25,4) počet automorfiz-

mov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$, ktorý ale nemôžme porovnať s parametrom optimálneho kódu $o(n,m,d)$, pretože pre dĺžku kódu $n=36$ tento údaj nie je známy.

6.1.21 Cage(7,5)



Obr. 6.33: Cage(7,5)

Cage(7,5)		C(175,126,5)	
$ \text{ver} $	50	d	5
$ \text{edg} $	175	m	85071×10^{33}
H	50×175	acrshortG	126×175
$t(H)$	0,078s	$t(G)$	0,22s
$ \text{AutGroup}(\text{Cage}) $	252000	$ \text{AutGroup}(C) $	252000
$M(k,g)$	50	$p(n,m,d)$	$2,736 \times 10^{-11}$
R	$5/7$	$o(n,m,d)$	-

Tabuľka 6.20: Tabuľka Cage(7,5)

V tabuľke Cage(7,5) môžme vidieť, že klietka má rovnaký počet vrcholov $|\text{ver}|$ ako Moorove ohraničenie $M(7,5)$. Na základe počtu vrcholov $|\text{ver}|$ a

hrán $|\text{edg}|$ bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,078s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R . Z incidenčnej matice klietky H sme vedeli určiť úplný lineárny kód $C(175,126,5)$, ktorý má počet kódových slov $m=85071 \times 10^{33}$. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 0,22s. Takisto bolo možné určiť z lineárneho kódu $C(175,126,5)$ počet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$, ktorý ale nemôžme porovnať s parametrom optimálneho kódu $o(n,m,d)$, pretože pre dĺžku kódu $n=175$ tento údaj nie je známy.

6.1.22 Cage(7,7)

Cage(7,7)		C(2240,1601,7)	
$ \text{ver} $	640	d	7
$ \text{edg} $	2240	m	88925×10^{476}
H	640×2240	G	1601×2240
$t(H)$	7,476s	$t(G)$	39,238s
$ \text{AutGroup}(\text{Cage}) $	320	$ \text{AutGroup}(C) $	320
$M(k,g)$	302	$p(n,m,d)$	$8,212 \times 10^{-184}$
R	5/7	$o(n,m,d)$	-

Tabuľka 6.21: Tabuľka Cage(7,7)

V tabuľke Cage(7,7) môžme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov $|\text{ver}|$ ako Moorove ohraničenie $M(7,7)$. Na základe počtu vrcholov $|\text{ver}|$ a hrán $|\text{edg}|$ bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 7,476s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a infor-

mačný pomer R. Z incidenčnej matice klietky H sme vedeli určiť lineárny kód C(2240,1601,7), ktorý má počet kódových slov $m=88925 \times 10^{476}$. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 39,238s. Takisto bolo možné určiť z lineárneho kódu C(2240,1601,7) počet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$, ktorý ale nemôžme porovnať s parametrom optimálneho kódu $o(n,m,d)$, pretože pre dĺžku kódu $n=2240$ tento údaj nie je známy.

6.1.23 Cage(7,8)

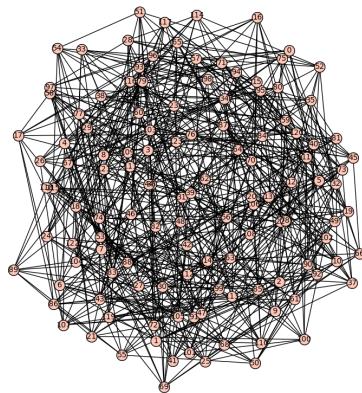
Cage(7,8)		C(2352,1681,8)	
ver	672	d	8
edg	2352	m	10750×10^{502}
H	672×2352	G	1681×2352
t(H)	8,513s	t(G)	43,11s
 AutGroup(Cage) 	14112	 AutGroup(C) 	14112
M(k,g)	518	p(n,m,d)	$2,213 \times 10^{-193}$
R	5/7	o(n,m,d)	-

Tabuľka 6.22: Tabuľka Cage(7,8)

V tabuľke Cage(7,8) môžme vidieť, že klietka nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(7,8). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 8,513s. Takisto bolo možné určiť z klietky počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R. Z incidenčnej matice klietky H sme vedeli určiť lineárny kód C(2352,1681,8), ktorý má počet kódových slov $m=10750 \times 10^{502}$. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo

trvalo 43,11ss. Takisto bolo možné určiť z lineárneho kódu C(2352,1681,8) počet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z klietky $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$, ktorý ale nemôžme porovnať s parametrom optimálneho kódu $o(n,m,d)$, pretože pre dĺžku kódu $n=2352$ tento údaj nie je známy.

6.1.24 Rec(10,5)



Obr. 6.34: Rec(10,5)

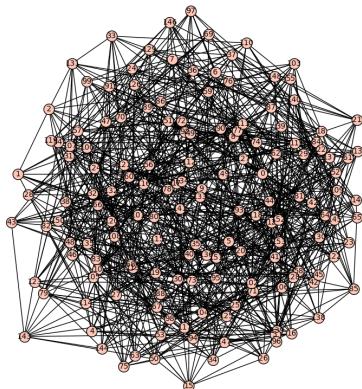
Rec(10,5)		C(620,497,5)	
ver	124	d	5
edg	620	m	40918×10^{144}
H	124×620	G	497×620
t(H)	0,44s	t(G)	2,879s
AutGroup(Cage)	1	AutGroup(C)	1
M(k,g)	101	p(n,m,d)	$1,810 \times 10^{-32}$
R	4/5	o(n,m,d)	-

Tabuľka 6.23: Tabuľka Rec(10,5)

V tabuľke Rec(10,5) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(10,5). Na základe počtu vr-

cholov $|ver|$ a hrán $|edg|$ bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,44s. Takisto bolo možné určiť z rekordného grafu počet automorfizmov $|\text{AutGroup}(\text{Cage})|$ a informačný pomer R . Z incidenčnej matice rekordného grafu H sme vedeli určiť lineárny kód $C(620,497,5)$, ktorý má počet kódových slov $m=40918 \times 10^{144}$. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 2,879s. Takisto bolo možné určiť z lineárneho kódu $C(620,497,5)$ počet automorfizmov $|\text{AutGroup}(C)|$, ktorý je zhodný s počtom automorfizmov z rekordného grafu $|\text{AutGroup}(\text{Cage})|$. Určili sme aj parameter perfektného $p(n,m,d)$, ktorý ale nemôžme porovnať s parametrom optimálneho kódu $o(n,m,d)$, pretože pre dĺžku kódu $n=620$ tento údaj nie je známy.

6.1.25 Rec(11,5)



Obr. 6.35: Rec(11,5)

Rec(11,5)		C(847,694,5)	
ver	154	d	5
edg	847	m	82190×10^{204}
H	154×847	G	694×847
t(H)	0,778s	t(G)	5,221s
AutGroup(Cage)	1	AutGroup(C)	1
M(k,g)	122	p(n,m,d)	$3,145 \times 10^{-41}$
R	9/11	o(n,m,d)	-

Tabuľka 6.24: Tabuľka Rec(11,5)

V tabuľke Rec(11,5) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(11,5). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 0,778s. Takisto bolo možné určiť z rekordného grafu počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice rekordného grafu H sme vedeli určiť lineárny kód C(847,694,5), ktorý má počet kódových slov $m=82190 \times 10^{204}$. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 5,221s. Takisto bolo možné určiť z lineárneho kódu C(847,694,5) počet automorfizmov |AutGroup(C)|, ktorý je zhodný s počtom automorfizmov z rekordného grafu |AutGroup(Cage)|. Určili sme aj parameter perfektného p(n,m,d), ktorý ale nemôžme porovnať s parametrom optimálneho kódu o(n,m,d), pretože pre dĺžku kódu n=847 tento údaj nie je známy.

6.1.26 Rec(12,5)

Rec(12,5)		C(1218,1016,5)	
ver	203	d	5
edg	1218	m	70222×10^{301}
H	203×1218	G	1016×1218
t(H)	1,351s	t(G)	10,986s
AutGroup(Cage)	203	AutGroup(C)	203
M(k,g)	145	p(n,m,d)	$1,155 \times 10^{-55}$
R	5/6	o(n,m,d)	-

Tabuľka 6.25: Tabuľka Rec(12,5)

V tabuľke Rec(12,5) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(12,5). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 1,351s. Takisto bolo možné určiť z rekordného grafu počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice rekordného grafu H sme vedeli určiť lineárny kód C(1218,1016,5), ktorý má počet kódových slov $m=70222 \times 10^{301}$. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 10,986s. Takisto bolo možné určiť z lineárneho kódu C(1218,1016,5) počet automorfizmov |AutGroup(C)|, ktorý je zhodný s počtom automorfizmov z rekordného grafu |AutGroup(Cage)|. Určili sme aj parameter perfektného p(n,m,d), ktorý ale nemôžme porovnať s parametrom optimálneho kódu o(n,m,d), pretože pre dĺžku kódu n=1218 tento údaj nie je známy.

6.1.27 Rec(13,5)

Rec(13,5)		C(1495,1266,5)	
ver	230	d	5
edg	1495	m	12705×10^{377}
H	230×1495	G	1266×1495
t(H)	1,927s	t(G)	16,658s
AutGroup(Cage)	1	AutGroup(C)	1
M(k,g)	170	p(n,m,d)	$1,296 \times 10^{-63}$
R	11/13	o(n,m,d)	-

Tabuľka 6.26: Tabuľka Rec(13,5)

V tabuľke Rec(13,5) môžme vidieť, že rekordný graf nemá rovnaký ale väčší počet vrcholov |ver| ako Moorove ohraničenie M(13,5). Na základe počtu vrcholov |ver| a hrán |edg| bolo možné určiť očakávaný rozmer kontrolnej matice H a takisto samotnú maticu vytvoriť a uložiť, čo trvalo 1,927s. Takisto bolo možné určiť z rekordného grafu počet automorfizmov |AutGroup(Cage)| a informačný pomer R. Z incidenčnej matice rekordného grafu H sme vedeli určiť lineárny kód C(1495,1266,5), ktorý má počet kódových slov $m=12705 \times 10^{377}$. Vieme určiť rozmer generujúcej matice G ako aj samotnú maticu vytvoriť a uložiť, čo trvalo 16,658s. Takisto bolo možné určiť z lineárneho kódu C(1495,1266,5) počet automorfizmov |AutGroup(C)|, ktorý je zhodný s počtom automorfizmov z rekordného grafu |AutGroup(Cage)|. Určili sme aj parameter perfektného p(n,m,d), ktorý ale nemôžme porovnať s parametrom optimálneho kódu o(n,m,d), pretože pre dĺžku kódu n=1495 tento údaj nie je známy.

6.2 Výsledky

6.2.1 Zhrnutie výsledkov

V nasledovnej tabuľke si zobrazíme dosiahnuté výsledky a porovnáme ich s vypočítaným Moorovým ohraničením $M(k,g)$ pre povolený počet vrcholov $|ver|$ ako aj získané lineárne kódy $C(n,m,d)$ s perfektnými pomocou zadefinovaného parametra $p(n,m,d)$.

Cage(k,g)	M(k,g)	H	 Aut.(C) 	G	p(n,m,d)
Cage(3,5)	10	10×15	120	6×15	0,236328
Cage(3,6)	14	14×21	336	8×21	0,028320
Cage(3,7)	22	24×36	32	13×36	0,000931
Cage(3,8)	30	30×45	1440	16×45	0,000028
Cage(3,10)	62	70×105	80	36×105	-
Cage(3,11)	94	112×168	64	57×168	-
Rec(3,14)	254	384×576	96	193×576	-
Rec(3,16)	510	960×1440	96	481×1440	-
Rec(3,17)	766	2176×3264	544	1089×3264	-
Rec(3,18)	1022	2560×3840	640	1281×3840	-
Rec(3,20)	2046	5376×8064	2688	-	-
Rec(3,23)	6142	49326×73989	1	-	-
Rec(3,25)	12286	108906×163359	-	-	-

Tabuľka 6.27: Tabuľka dosiahnutých výsledkov č.1

Cage(k,g)	M(k,g)	H	Aut.(C)	G	p(n,m,d)
Cage(4,5)	17	19×38	24	20×38	0,002831
Cage(4,7)	53	67×134	4	68×134	$5,436 \times 10^{-15}$
Cage(4,9)	161	270×540	90	271×540	-
Cage(4,10)	242	384×768	768	385×768	-
Cage(5,10)	682	1296×3240	3888	1945×3240	-
Cage(6,4)	12	12×36	1036800	25×36	0,018066
Cage(7,5)	50	50×175	252000	126×175	$2,736 \times 10^{-11}$
Cage(7,7)	302	640×2240	320	1601×2240	$8,212 \times 10^{-184}$
Cage(7,8)	518	672×2352	14112	1681×2352	$2,213 \times 10^{-193}$
Rec(10,5)	101	124×620	1	497×620	$1,810 \times 10^{-32}$
Rec(11,5)	122	154×847	1	694×847	$3,145 \times 10^{-41}$
Rec(12,5)	145	203×1218	203	1016×1218	$1,155 \times 10^{-55}$
Rec(13,5)	170	230×1495	1	1266×1495	$1,296 \times 10^{-63}$

Tabuľka 6.28: Tabuľka dosiahnutých výsledkov č.2

6.2.2 Vyhodnotenie výsledkov

Vo všetkých prípadoch bola splnená podmienka Moorovho ohraničenia $M(k,g)$ zaručujúca existenciu uvažovaných klietok Cage(k,g) alebo rekordných grafov Rec(k,g). Vo všetkých prípadoch sa nám podarilo zistiť rozmer kontrolnnej (incidenčnej) matice H, počet automorfizmov $|\text{AutGroup}(C)|$ až na rekordný graf rec(3, 25), kedy program nedokázal vypočítať počet automorfizmov $|\text{AutGroup}(C)|$. Správnosť $|\text{AutGroup}(C)|$ sme overili pomocou hodnôt

$|\text{AutGroup}(C)|$ z tabuľky rekordných grafov ako aj z informácií o známych klietkach, kde je tento údaj pre niektoré uvažované klietky $\text{Cage}(k,g)$ alebo rekordné grafy $\text{Rec}(k,g)$ známy ako aj údaj o počte vrcholov $|ver|$. Na zaklade incidenčnej matice klietky $\text{Cage}(k,g)$ sme boli schopní vygenerovať lineárny kód $C(n,m,d)$ ako aj jeho generujúcu maticu G až na prípady $cage(3, 20)$, $cage(3, 23)$ a $cage(3, 25)$, kedy už bolo výpočtovo náročné generujúcu maticu G vygenerovať. Z generujúcich matíc G bolo možné zistiť maximálny počet kódových slov m v kóde a minimálnu kódovú vzdialenosť d , pre klietky $g \leq 8$. Pre lineárne kódy, ktoré vznikli z väčších obvodov ako osem bolo príliž výpočtovo náročné zistiť minimálnu kódovú vzdialenosť d a takisto parameter perfektného kódu $p(n,m,d)$, ktorý vyjadruje vzdialenosť nášho kódu od perfektného kódu (v intervale $<0,1>$, kedy 1 znamená prefektný kód). Tento parameter sme porovnávali s parametrom optimálneho kódu $o(n,m,d)$ v dvoch prípadoch $cage(3, 5)$, $cage(3, 6)$, kedy to bolo možné a zistili sme, že hodnoty sa veľmi mierne líšili. Sledovali sme aj čas generovania kontrolných $t(H)$ a generujúcich matíc $t(G)$, ktorý naznačoval výpočtovú náročnosť.

Kapitola 7

Záver

XXXXXX

References

- [Hil88] R. Hill. *A First Course in Coding Theory*. Zv. 72-74. Oxford applied mathematics and computing science series 459. Oxford University Press, 1988, s. 251. ISBN: 0-19-853803-0. DOI: 10.2307/3618021.
- [HP03] W Cary Huffman a Vera Pless. *Fundamentals of Error-Correcting Codes, Basic concepts of linear codes*. Cambridge Univ. Press, 2003, s. 1–20. ISBN: 9780511807077. DOI: <https://doi.org/10.1017/CBO9780511807077>.
- [Mal07] G.A. Malema. *Low-Density Parity-Check Codes: Construction and Implementation. Dissertation Thesis*. School of Electrical, Electronic Engineering, Faculty of Engineering, Computer a Mathematical Sciences The University of Adelaide, Australia, 2007.
- [Gos12] Sharma Rakesh Goswami Ashish. “Low Complexity, High Performance LDPC Codes Based on Defected Fullerene Graphs”. In: *International Journal of Mathematical and Computational Sciences* 6.2 (2012), s. 136–138. DOI: 10.5281.
- [EJ13] Geoffrey Exoo a Robert Jajcay. “Dynamic cage survey”. In: *The Electronic Journal of Combinatorics* DS16.3 (2013), s. 55. DOI: 10.37236/37.

- [Mis+13] Rafael Misoczki et al. *MDPC-McEliece: New McEliece variants from Moderate Density Parity-Check codes*. IEEE, 2013, s. 1–21.
ISBN: 978-1-4799-0446-4. DOI: 10.1109/ISIT.2013.6620590.
- [Bor18] Branislav Boráň. *Hustota inverzií riedkych cyklických matíc. Bachelor's Thesis*. Slovenská technická univerzita v Bratislave, Fakulta elektrotechniky a informatiky, Slovenská republika, 2018.
- [Bro18] Andries Brouwer. *Small binary codes*. Available at: <https://www.win.tue.nl/~aeb/codes/binary-1.html>. 2018.
- [Nas19] Nisreen Mohammad Nashash. *On Then Automorphism Groups of Some Linear Codes. Master's Thesis*. Faculty of Graduate Studies, Hebron University, Palestine, 2019.
- [Sag20] Inc. Sagemath. *CoCalc – Collaborative Calculation and Data Science*. Available at: <https://cocalc.com>. 2020.
- [The+20] W. Stein The Sage Developers et al. *SageMath, version 9.0*. Available at: <http://www.sagemath.org>. 2020.
- [Exo] Geoffrey Exoo. *Regular Graphs of Given Degree and Girth*. Available at: <http://cs.indstate.edu/ge/CAGES/index.html>.

Zoznam skratiek

|AutGroup(C)| Počet automorfizmov získaný z lineárneho kódu. 15, 24, 28, 35–37, 39–51, 53–58, 60–68, 70, 71

|AutGroup(Cage)| Počet automorfizmov získaný z klietky. 14, 15, 18, 22, 23, 27, 28, 35–37, 39–51, 53–58, 60–68

|cols| Počet stĺpcov matice. 9

|edg| Počet hrán grafu. 22, 23, 35, 37, 39, 41–51, 53–58, 60–68

|rows| Počet riadkov matice. 9

|ver| Počet vrcholov grafu. 11, 13–18, 20–23, 27, 35, 37, 39, 41–51, 53–58, 60–69, 71

AutGroup(C) Grupa automorfizmov získaná z lineárneho kódu. 18, 24

AutGroup(Cage) Grupa automorfizmov získaná z klietky. 13

C(n,m,d) Lineárny kód. 5, 6, 9, 10, 17, 19, 23–25, 27, 28, 33, 34, 69, 71

C(n,r) Lineárny kód. viii, 4–6

Cage(k,g) Klietka. 13–15, 17, 18, 20–23, 27, 28, 31, 33, 69–71

d Minimálna Hammingova vzdialenosť v kóde. viii, 5–9, 24, 25, 27, 35, 37, 39, 41–51, 53, 54, 56–58, 60–64, 66–68, 71, 78

dim(C) Dimenzia pod priestoru lineárneho kódu. 5

F(2) Priestor n rozmernych vektorov. 4–6, 10

G Generujúca matica lineárneho kódu. viii, 5, 6, 24–26, 28, 32, 35–37, 39, 41–51, 53–58, 60, 62–71, 80, 81

g Veľkosť obvodu grafu. viii, 11, 13, 14, 17, 21, 22

H Kontrolná matica lineárneho kódu, Incidenčná matica klietky. 5, 6, 9–13, 17, 18, 23–28, 32, 35–70, 80, 81

k Stupeň vrcholov v grafe. 11, 13, 14, 21, 22

LDPC Low density parity check codes - Lineárne kódy s malou hustotou matice H. viii, 10, 11, 13

m Počet slov v kóde. viii, 5, 6, 10, 24, 25, 27, 35, 37, 39, 41–51, 53–58, 60–68, 71

M(k,g) Moorové ohraničenie pre klietku. 13, 21, 22, 27, 29, 35, 37, 39, 41, 42, 44–51, 53, 54, 56–58, 60–64, 66–70

n Dĺžka lineárneho kódu. viii, 4–8, 10, 25, 40, 42, 54, 55, 61–68, 78

o(n,m,d) Parameter optimálneho kódu. 25, 28, 35–37, 39–51, 53–58, 60–68, 71

p(n,m,d) Parameter perfektného kódu. 25, 28, 31, 35–37, 39–51, 53–58, 60–71

QC-MDPC Kvázicyklické lineárne kódy s väčšou hustotou matice H ako LDPC. 10, 11

R Informačný pomer. viii, 9, 18, 22, 23, 27, 35, 37, 39, 41–51, 53–58, 60–68

r Počet lineárne nezávislých vektorov, Dimenzia pod priestoru lineárneho kódu. 4–6, 24

Rec(k,g) Rekordný graf. 16–18, 20–23, 27, 28, 70, 71

t(G) Čas generovania generujúcej matice lineárneho kódu. 28, 35, 37, 39, 41, 42, 44–51, 53, 54, 56–58, 60–64, 66–68, 71

t(H) Čas generovania kontrolnej matice lineárneho kódu. 28, 35, 37, 39, 41, 42, 44–51, 53, 54, 56–58, 60–64, 66–68, 71

V(n,2) Množina všetkých usporiadaných n-tíc F(2). 4–6

Zoznam tabuliek

3.1	Tabuľka známych $A_2(n, d)$ č.1	7
3.2	Tabuľka známych $A_2(n, d)$ č.2	8
3.3	Tabuľka známych Rec(k,g)	16
6.1	Tabuľka Cage(3,5)	35
6.2	Tabuľka Cage(3,6)	37
6.3	Tabuľka Cage(3,7)	39
6.4	Tabuľka Cage(3,8)	41
6.5	Tabuľka Cage(3,10)	42
6.6	Tabuľka Cage(3,11)	44
6.7	Tabuľka Rec(3,14)	45
6.8	Tabuľka Rec(3,16)	46
6.9	Tabuľka Rec(3,17)	47
6.10	Tabuľka Rec(3,20)	48
6.11	Tabuľka Rec(3,20)	49
6.12	Tabuľka Rec(3,23)	50
6.13	Tabuľka Rec(3,25)	51
6.14	Tabuľka Cage(4,5)	53
6.15	Tabuľka Cage(4,7)	54
6.16	Tabuľka Cage(4,9)	56

ZOZNAM TABULIEK 79

6.17 Tabuľka Cage(4,10)	57
6.18 Tabuľka Cage(5,10)	58
6.19 Tabuľka Cage(6,4)	60
6.20 Tabuľka Cage(7,5)	61
6.21 Tabuľka Cage(7,7)	62
6.22 Tabuľka Cage(7,8)	63
6.23 Tabuľka Rec(10,5)	64
6.24 Tabuľka Rec(11,5)	66
6.25 Tabuľka Rec(12,5)	67
6.26 Tabuľka Rec(13,5)	68
6.27 Tabuľka dosiahnutých výsledkov č.1	69
6.28 Tabuľka dosiahnutých výsledkov č.2	70

Zoznam obrázkov

3.1	Vzťah medzi grafom a kontrolnou maticou [Mal07]	12
3.2	Petersenov graf (vľavo) a Heawoodov graf (vpravo) [EJ13] . .	14
3.3	McGeeho graf (vľavo) a Tutteov-Coxeterov graf (vpravo)[EJ13]	15
5.1	Sage a Cocalc[sagemath][Sag20]	19
6.1	M(k,g) - riešenie	29
6.2	Konštrukcia klietky Cage(k,g) - riešenie	29
6.3	Konštrukcia Cage(6,4) - riešenie	30
6.4	Parser zoznamu susednosti - riešenie	30
6.5	Validácia klietok - riešenie	31
6.6	Parameter perfektného kódu - riešenie	31
6.7	Uloženie matice H do textoveho suboru - riešenie	32
6.8	Získanie matice H a uloženie matice G do textového súboru - riešenie	32
6.9	Validácia lineárneho kódu - riešenie	33
6.10	Počet automorfizmov z klietky - riešenie	33
6.11	Počet automorfizmov z lineárneho kódu - riešenie	34
6.12	Cage(3,5) [sagemath]	34
6.13	H_Cage(3,5) (vľavo) a G_Cage(3,5) (vpravo)	35
6.14	Cage(3,6) [sagemath]	36

6.15 H_Cage(3,6) (vľavo) a G_Cage(3,6) (vpravo)	36
6.16 Cage(3,7) [sagemath]	38
6.17 H_Cage(3,7)	38
6.18 G_Cage(3,7)	39
6.19 Cage(3,8) [sagemath]	40
6.20 H_Cage(3,8)	40
6.21 G_Cage(3,8)	41
6.22 Cage(3,10) [sagemath]	42
6.23 Cage(3,11) [sagemath]	43
6.24 Rec(3,14)	45
6.25 Cage(4,5) [sagemath]	52
6.26 H_Cage(4,5)	52
6.27 G_Cage(4,5)	53
6.28 Cage(4,7)	54
6.29 Cage(4,9)	55
6.30 Cage(6,4)	59
6.31 H_Cage(6,4)	59
6.32 G_Cage(6,4)	60
6.33 Cage(7,5)	61
6.34 Rec(10,5)	64
6.35 Rec(11,5)	65