UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KONŠTRUKTÍVNA ENUMERÁCIA GRAFOV

DIPLOMOVÁ PRÁCA

2015 Bc. Anton Kovaľ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KONŠTRUKTÍVNA ENUMERÁCIA GRAFOV

DIPLOMOVÁ PRÁCA

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Vedúci diplomovej práce práce:	doc. RNDr. Martin Mačaj, PhD.

Bratislava, 2015 Bc. Anton Kovaľ

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Anton Kovaľ
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Konštruktívna enumerácia grafov
Constructive enumeration of graphs

Cieľ: Naštudovať techniky konštruktívnej enumerácie grafov a aplikovať ich na problém nájdenia všetkých malých silne regulárnych grafov s automorfizmom rádu 3.

Vedúci: doc. RNDr. Martin Mačaj, PhD.
Katedra: FMFI.KAGDM - Katedra algebry, geometrie a didaktiky matematiky
Vedúci katedry: prof. RNDr. Pavol Zlatoš, PhD.

Dátum zadania: 23.10.2013

Dátum schválenia: 26.11.2013 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie

Chcel by som sa poďakovať môjmu vedúcemu doc. RNDr. Martinovi Mačajovi, PhD. za výber témy, jeho pomoc, inšpiratívne rady a dohľad nad mojou činnosťou. Tiež by som chcel poďakovať svojim rodičom za podporu počas štúdia a M. Kovaľovi za poskytnutie počítačového času na výpočty.

Abstrakt

Silne regulárny graf s parametrami (n, k, λ, μ) je k-regulárny graf rádu n, v ktorom každá dvojica susedných vrcholov má práve λ spoločných susedov a každá dvojica nesusedných vrcholov má práve μ spoločných susedov. Pre sady parametrov (n, k, λ, μ) existuje niekoľko nutných podmienok, ktoré pripúšťajú existenciu silne regulárnych grafov. Sadu parametrov, ktorá spĺňa všetky podmienky budeme nazývať prípustné. Aj keď existencia silne regulárnych grafov je známa pre všetky prípustné sady parametrov s $n \leq 55$, kompletná klasifikácia pre sady parametrov (37, 18, 8, 9), (41, 20, 9, 10), (45, 22, 10, 11), (49, 24, 11, 12), (49, 18, 7, 6), (50, 21, 8, 9) a (53, 26, 12, 13) nie je známa. Témou diplomovej práce je využitie konštruktívnej enumerácie, ktorej cieľom je nájsť všetky objekty s rovnakými vlastnosťami, ktoré nie sú izomorfné medzi sebou. Úplným prehľadávaním pomocou počítača sme získali zoznam všetkých silne regulárnych grafov do 55 vrcholov, ktoré majú automorfizmus rádu 3.

KĽÚČOVÉ SLOVÁ: konštruktívna enumerácia grafov, silne regulárne grafy

Abstract

A k-regular graph of order n is said to be strongly regular with parameters (n, k, λ, μ) if each pair of adjacent vertices has exactly λ common neighbors and each pair of non-adjacent vertices has exactly μ common neighbors. There several necessary conditions exist for parameter sets to admit the existence of a strongly regular graph. Parameter sets satisfying all of them are called feasible. Although the existence of a strongly regular graph is solved for all feasible parameter sets with $n \leq 55$, the complete classification is still open for parameter sets $(37, 18, 8, 9)$, $(41, 20, 9, 10)$, $(45, 22, 10, 11)$, $(49, 24, 11, 12)$, $(49, 18, 7, 6)$, $(50, 21, 8, 9)$ and $(53, 26, 12, 13)$. The topic of diploma thesis is using constructive enumeration on strongly regular graphs. Main goal of constructive enumeration is to find all objects with some specific properties up to isomorphism. By exhaustive computer search we obtain a full list of strongly regular graphs on up to 55 vertices with an automorphism of order 3.

KEYWORDS: constructive enumeration of graphs, strongly regular graph

OBSAH

Úvod

Konštruktívna enumerácia grafov je postup na nájdenie všetkých grafov s danými vlastnosťami. Cieľom konštruktívnej enumerácie objektov je nájsť všetky objekty s rovnakými vlastnosťami, ktoré nie sú izomorfné medzi sebou. Hlavnou myšlienkou je, že pre každú triedu objektov vzhľadom na izomorfizmus vygenerujeme jedného reprezentanta. Novo vygenerované objekty testujeme na izomorfizmus s reprezentantmi.

Zaujímavou triedou grafov sú silne regulárne grafy so špecifickými algebraickými vlastnosťami, vďaka čomu ich využitie môžeme nájsť v rôznych oblastiach chémie [Ste14], teórii grúp [Bab14] alebo kvantovej fyziky [JJ14]. Práca je rozdelená do 6 kapitol. V prvej kapitole definujeme základné pojmy z teórie grafov a teórie grúp, ktoré v práci používame. V ďalšej kapitole sa venujeme silne regulárnym grafom a ich vlastnostiam. Potom nasleduje prehľadová kapitola o konštruktívnej enumerácii kombinatorických objektov, kde preberáme základné techniky. V štvrtej kapitole definujeme orbitné matice a ukážeme ich vlastnosti ako boli prezentované *M. Behbahanim* a *C. Lamom* v dizertačnej práci [Beh09]. V ďalšej kapitole sa zaoberáme len orbitnými maticami s automorfizmom rádu 3, ktorými sa predchádzajúci autori veľmi nezaoberali. Na základe ich ideí sme pre tieto matice navrhli vlastnú implementáciu v programovacom jazyku *Scala* [sca15a]. Záverečná kapitola obsahuje dosiahnuté výsledky a ich porovnanie s už známymi grafmi.

Hlavným výsledkom práce je vytvorenie programu na hľadanie potencionálnych orbitných matíc silne regulárnych grafov s automorfizmom rádu 3. Pomocou tohoto programu sme vytvorili úplný zoznam potencionálnych orbitných matíc silne regulárnych grafov s najviac 55 vrcholmi a parametrami, pre ktoré nie je známa úplna klasifikácia. Z týchto orbitných matíc boli následne vytvorené úplné zoznamy silne regulárnych grafov do 55 vrcholov s automorfizmom rádu 3 ako aj podstatne rozšírené zoznamy známych silne regulárnych grafov do 55 vrcholov, resp. regulárnych 2-grafov do 60 vrcholov.

Kapitola 1

Základné definície a vety

V tejto kapitole sú zhrnuté základné definície a vety, ktoré v práci používame. Prvá časť obsahuje základnú terminológiu z teórie grafov. V druhej časti sú uvedené základné vlastnosti matice susednosti a tvrdenia z lineárnej algebry, ktoré neskôr v práci využívame. V tretej časti sú uvedené základné definície a vety z teórie grúp.

1.1 Teória grafov

Základná terminológia z teórie grafov je intuitívna, vychádzame zo štandardnej notácie.

Graf je usporiadaná dvojica $G = (V, E)$ množín spĺňajúca $E \subseteq \binom{V}{2}$, t.j. prvky z E sú dvojprvkové podmnožiny V. Prvky z množiny V nazývame *vrcholy* grafu G a prvky z množiny E nazývame *hrany*. Pri zakreslení grafu bod predstavuje vrchol a čiara medzi bodmi predstavuje hranu.

Počet vrcholov grafu G určuje *veľkosť grafu*. Vrchol v je *incidentný* s hranou e ak $v \in e$. Stupeň vrchola v je počet hrán z E incidentných s v. Ak všetky vrcholy G majú rovnaký stupeň k, potom graf G je k-regulárny. Vrcholy, ktoré sú spojené hranou nazývame *susedné*.

Kompletný graf je graf, v ktorom je každý vrchol grafu spojený hranou s každým iným vrcholom grafu. Kompletný graf s n vrcholmi označujeme K_n.

Nech $G = (V, E)$ a $G' = (V', E')$ sú dva grafy. Bijekciu $\varphi : V \to V'$ takú, že $xy \in E \Leftrightarrow \varphi(x)\varphi(y) \in E'$ nazývame *izomorfizmus* a grafy G a G' sú *izomorfné*, ozn. $G \simeq G'$. Izomorfizmus grafu G na seba nazývame *automorfizmus* grafu G.

Graf $G' = (V', E')$ je *podgraf* grafu $G = (V, E)$, ak platí $V' \subseteq V$ a $E' \subseteq E$, označujeme $G' \subseteq G$. Ak $G' \subseteq G$ a zároveň graf G' obsahuje všetky hrany $xy \in E$ s $x, y \in V'$, potom graf

G' nazývame *indukovaný podgraf* grafu G. *Komplement* $\overline{G}$ grafu G je graf na V s množinou hrán $\binom{V}{2} - E$.

Sled v grafe G, presnejšie $u - v$ sled je striedavá postupnosť vrcholov a hrán tvaru $u = v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n = v$, kde $e_i = v_{i-1}v_i$. Dĺžku sledu nám určuje číslo n. Ak vrchol u je totožný s vrcholom v, potom $u - v$ sled nazývame *uzavretým sledom*.

Cesta v grafe G je sled, v ktorom sa neopakujú ani vrcholy, ani hrany. *Vzdialenosť* dvoch vrcholov v grafe G je dĺžka najkratšej cesty medzi nimi. *Súvislý graf* je graf, v ktorom existuje cesta z každého vrcholu do každého vrcholu. *Komponent grafu* G je taký súvislý podgraf grafu G, ktorý nie je obsiahnutý v žiadnom väčšom súvislom podgrafe grafu G, t.j. maximálne súvislý podgraf.

Nech *Kroneckerova delta funkcia* je δ_{ij} definovaná takto:

$$\delta_{ij} = \begin{cases} 1 & \text{ak } i = j \\ 0 & \text{inak} \end{cases}$$

Matica susednosti grafu G s n vrcholmi je štvorcová matica $A = A(G) = (a_{i,j})$ stupňa n, ktorej riadky aj stĺpce sú indexované vrcholmi grafu. Pre prvok matice A platí: $a_{i,j} = 1$ práve vtedy, keď vrcholy v_i a v_j sú susedné v grafe G. V opačnom prípade je $a_{i,j} = 0$.

Matica susednosti A pre graf G je symetrická a na diagonále má samé nuly.

Vlastné čísla grafu G sú vlastné čísla jeho matice susednosti A. Ak preusporiadame vrcholy v G (priradíme im čísla $1, 2, 3, \ldots, n$ v inom poradí) dostaneme maticu susednosti PAP^{-1}, kde P je nejaká permutačná matica. Z lineárnej algebry vyplýva, že vlastné čísla grafu nezávisia od usporiadania vrcholov, resp. *spektrum matice* (množina vlastných čísel) sa nemení.

Ak G je k-regulárny graf, potom jednoduchou úvahou dostaneme rovnosť:

$$AJ = JA = kJ, \tag{1.1}$$

kde štandardne matica J (all-one matrix) označuje maticu, ktorá obsahuje samé jednotky. Z tejto rovnici vyplýva, že matica A má vlastné číslo k s násobnosťou aspoň 1.

Existuje zaujímavý vzťah medzi sledmi v grafe a maticou susednosti, ktorý sa dá ľahko dokázať pomocou matematickej indukcie.

Lema 1.1. *Nech A je matica susednosti grafu G a nech u, v sú vrcholy grafu G. Potom počet $u - v$ sledov s dĺžkou m je rovný $(A^m)_{u,v}$. Počet uzavretých sledov dĺžky m v grafe G je rovný $Tr(A^m)$.*

1.2 Spektrálne vlastnosti matice susednosti

V tejto časti sú zhrnuté základné vlastnosti matice susednosti pre jednoduchý neorientovaný graf, ktoré vyplývajú z lineárnej algebry. Plné znenia definícii a tvrdení z lineárnej algebry je možné nájsť v [Zla11].

Reálna symetrická matica je ortogonálne podobná s (reálnou) diagonálnou maticou, z čoho vyplývajú nasledovné dôsledky.

Matica susednosti A má práve n vlastných čísel, pričom všetky vlastné čísla sú reálne. Hodnosť $h(A)$ matice susednosti A je rovná počtu nenulových vlastných čísel. Ku každému vlastnému číslu existuje vlastný vektor matice A.

Vlastné vektory α a β s rôznymi vlastnými číslami matice susednosti A sú ortogonálne (kolmé). Z vlastných vektorov matice susednosti A vieme zostrojiť ortonormálnu bázu.

Nech $\lambda_1, \ldots, \lambda_n$ sú vlastné čísla matice susednosti A, potom platí:

$$tr(A) = \sum_i \lambda_i \tag{1.2}$$

1.3 Teória grúp

V tejto časti sú zhrnuté základné definície, označenia a vety z teórie grúp. Dôkazy ku tvrdeniam je možné nájsť v [Zla11], [Ser03] a [Tom11].

Grupa $(G, \circ)$ je neprázdna množina G s binárnou asociatívnou operáciou $\circ$, ktorá má neutrálny prvok e a v ktorom ku každému prvku a existuje inverzný prvok a^{-1}.

Neskôr v práci budeme používať označenie ab namiesto $a \circ b$, ak binárna operácia je zrejmá z kontextu. Grupa G je *konečná*, ak G je konečná množina. Budeme sa zaoberať len konečnými grupami. Veľkosť $|G|$ množiny G nazývame *rád* grupy G. Keď G je konečná grupa, potom pre každý prvok $a \in G$ vždy existuje najmenšie kladné číslo m také, že $a^m = a \circ a \ldots \circ a = e$. Číslo m sa nazýva *rádom prvku* a. Nech S je podmnožina G. S *generuje* G, ak každý prvok $a \in G$ môže byť vyjadrený ako $a = b_1 \circ b_2 \circ \ldots \circ b_m$, kde $b_1, b_2, \ldots, b_m \in S$ pre nejaké m, ktorý zavisí len na prvku a. Potom S nazývame *množinu generátorov* G, ozn. $G = \langle S \rangle$. Prvky množiny S nazývame *generátory* grupy G. Štruktúrne najjednoduchšími grupami sú *cyklické grupy*, t.j. grupy generované jediným generátorom.

Permutácia je bijekcia z neprázdnej konečnej množiny X do seba samej. Nech π je permutácia množiny X a nech x^π označuje *obraz prvku* $x \in X$ v π. *Zloženie* dvoch permutácií

π a σ množiny X budeme označovať $\pi\sigma$ a zároveň platí $x^{\pi\sigma} = (x^{\pi})^{\sigma}$. *Symetrická grupa* na neprázdnej množine X je grupa, ktorej nosná množina je množina všetkých permutácii množiny X, a ktorej binárna operácia je skladanie permutácií. Symetrickú grupu na X budeme označovať $Sym(X)$ alebo $Sym(n)$, ak $X = \{1, \ldots, n\}$. Rád grupy $Sym(n)$ je $|Sym(n)| = n!$. *Permutačná grupa* je grupa G, ktorej prvky sú permutácie množiny X a ktorej grupová operácia je skladanie permutácií v G.

Nech $x \in X$ a $\pi \in Sym(X)$, potom π *stabilizuje* x, ak $x^{\pi} = x$. Dve permutácie $\pi, \sigma \in Sym(X)$ nazývame *disjunktné* práve vtedy, keď každý prvok z $x \in X$ buď π stabilizuje x a σ zobrazuje na x alebo naopak. Nech $x_1, x_2, \ldots, x_r \in \{1, \ldots, n\}$ sú rôzne. Permutáciu $\pi \in Sym(n)$ s obrazmi $x_1^{\pi} = x_2, x_2^{\pi} = x_3, \ldots, x_{r-1}^{\pi} = x_r, x_r^{\pi} = x_1$ s ďalšími, ktoré stabilizujú všetky ostatné čísla nazývame *r-cyklus*, ozn. $(x_1 x_2 \ldots x_r)$. Každá permutácia $\pi \in Sym(n)$ je buď identita alebo zloženie jedného alebo viacerých disjunktných cyklov.

Nech $(G, \circ)$ je grupa a nech $H \subset G$, potom $(H, \circ)$ je *podgrupa* grupy $(G, \circ)$, ak $(H, \circ)$ je grupa. Ak binárny operátor $\circ$ je zrejmý z kontextu, potom H podgrupu grupy G budeme označovať $H \leq G$. Ak $H \neq G$, potom H je *vlastná podgrupa* grupy G, ozn. $H \lneq G$. Nech G je grupa, $H \leq G$ a $g \in G$. *Pravou triedou* grupy G podľa H nazývame množinou $H \circ g = \{h \circ g : h \in H\}$. Podobne *ľavú triedu* grupy grupy G podľa H nazývame množinu $g \circ H = \{g \circ h : h \in H\}$. Rozklad grupy G podľa H je množina $G/H = \{g \circ H; g \in G\}$. Nech G je grupa, $H \leq G$. Potom všetky prvky množiny G/H majú rovnako veľa prvkov. Počet všetkých ľavých (pravých) tried rozkladu G podľa H nazývame *indexom grupy* grupy G podľa H, ozn. $[G : H]$.

Veta 1.2 (Lagrange)**.** *Nech G je konečná grupa a $H \leq G$. Potom $|H|$ delí $|G|$ a $[G : H] = |G| / |H|$.*

Nech G je grupa, $H \leq G$. Podmnožina $T \subseteq G$ je *pravá (ľavá) transverzála* H v G, ak T obsahuje práve jeden prvok z každej pravej (ľavej) triedy grupy G podľa H. Prvky z T nazývame *reprezentantmi* triedy rozkladu. Je zrejmé, že G sa rovná zjednotení všetkých disjunktných pravých tried Ht, kde $t \in T$. Zároveň ak $g \in G$, potom existuje práve jeden reprezentant $t \in T$ a práve jeden prvok $h \in H$, pre ktoré platí $g = ht$.

Nech $(G, \circ)$ je grupa a X je neprázdna množina. Binárnu operáciu $\cdot$, $G \times X \to X$, nazývame *ľavú akciu* grupy G na množine X, ak $a \cdot x \in X$, $e \cdot x = x$ a $(b \circ c) \cdot x = b \cdot (c \cdot x)$ pre všetky $a, b, c \in G$ a $x \in X$. Podobne *pravá akcia* grupy G na množine X, ak $x \cdot a \in X$, $x \cdot e = x$ a $x \cdot (b \circ c) = (x \cdot b) \cdot c$ pre všetky $a, b, c \in G$ a $x \in X$. Ďalej v práci budeme

používať označenie gx namiesto $g \cdot x$ a x^g namiesto $x \cdot g$, kde $x \in X, g \in G$, ak operácia $\cdot$ bude zrejmá z kontextu.

Pravá akcia permutačnej grupy $G \leq Sym(X)$ na X je definovaná $x \cdot g = x^g$ pre každé $x \in X$ a $g \in G$. Pre každé $x \in X$ množinu $x^G = \{x^g : g \in G\}$ nazývame *orbitou* x v G. Množinu všetkých orbít z X na pravú (ľavú) akciu grupy G budeme označovať $G \parallel X$. Pre každé $x \in X$ množinu $G_x = \{g \in G : x^g = x\}$ nazývame *stabilizátorom* x v G. Ľahko vidieť, že G_x je podgrupou grupy G.

Veta 1.3 (Orbit-Stabilizer)**.** *Nech grupa G má pravú (ľavú) akciu na množinu X a nech $x \in X$. Potom $|x^G| = |G : G_x|$.*

Predpokladajme symetrickú grupu $Sym(n)$ a akciu tejto grupy na konečnú množinu G_n, grafy s veľkosťou n. Potom vieme G_n definovať pomocou $\pi \cdot X$, ak platí $V(\pi \cdot X) = V(X)$ a zároveň $E(\pi \cdot X) = \{\{x, y\} : \{x^\pi, y^\pi\}\} \in E(X)\}$ pre každé $X \in G_n$ a $\pi \in Sym(n)$. Taktiež platí $(\pi\sigma) \cdot X = \pi \cdot (\sigma \cdot X)$ pre každé $X \in G_n$ a $\pi, \sigma \in Sym(n)$. Z toho vyplýva $V(\pi \cdot (\sigma \cdot X)) = V(\pi\sigma \cdot X)$ a $E(\pi \cdot (\sigma \cdot X)) = E(\pi\sigma \cdot X)$.

Dva grafy $G, H \in G_n$ pomocou akcie grúp na množine vrcholov grafov budeme nazývať *izomorfné* práve vtedy, keď existuje $\pi \in Sym(n)$ také že $\pi \cdot G = H$. Permutácia π predstavuje *izomorfizmus* z G do H.

Vo všeobecnom kontexte, pre konečnú množinu kombinatorických objektov Δ na množine $\{1, \ldots, n\}$ akcia grupy na Δ zodpovedá všetkým izomorfizmom medzi objektmi Δ. To znamená, že pre každý $g \in G$ a $X \in \Delta$ zodpovedá objekt $g \cdot X$ získaný akciou grupy g na X. Dve objekty X, Y sú *izomorfné*, ozn. $X \cong_G Y$ práve vtedy, keď existuje $g \cdot X = Y$. Ak grupa G je zrejmá z kontextu, budeme označovať namiesto $X \cong_G Y$ iba $X \cong Y$. Ak existuje $g \in G$ také, že $g \cdot X = X$, potom g nazývame *automorfizmom* množiny X. Množinou všetkých automorfizmov X nám vytvára *grupu automorfimov* množiny X, ktorú budeme označovať $Aut(X)$.

Pre množinu kombinatorických objektov Δ nám orbita $X \in \Delta$ v G zodpovedá množinou kombinatorických objektov Δ, ktoré sú izomorfné s X, t. j. $GX = \{g \cdot X : g \in G\} = \{Y \in \Delta : X \cong_G Y\}$. Stabilizátor X v G nám zodpovedá grupou automorfizmov množiny X, t.j. $G_X = \{g \in G : g \cdot X = X\} = Aut(X)$.

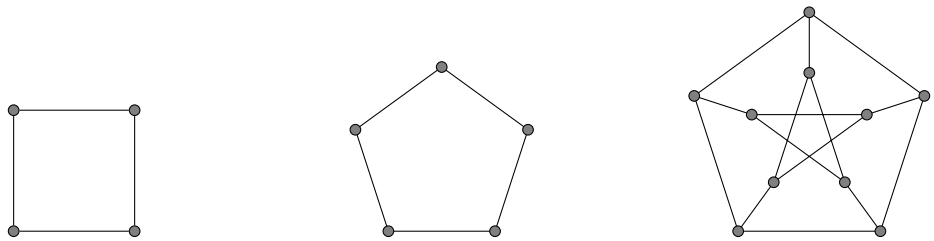
Kapitola 2

Silne regulárne grafy

V tejto kapitole sa zaoberáme silne regulárnymi grafmi, ktoré majú zaujímavé algebraické vlastnosti. Dôkazy k tvrdeniam a podrobnejšie informácie o silne regulárnych grafoch môžeme nájsť v [Kov13], ktorá vychádzala z [BH12].

2.1 Definícia

Definícia 2.1. *Graf G je silne regulárny graf (strongly regular graph) s parametrami (n, k, λ, μ), kde n je počet vrcholov, k určuje stupeň regulárnosti grafu, ľubovoľné dva susedné vrcholy grafu majú práve λ spoločných susedov a ľubovoľné dva nesusedné vrcholy majú práve μ spoločných susedov.*



(a) (b) (c)

Obrázok 2.1: Príklady malých silne regulárnych grafov: a) štvoruholník (4, 2, 0, 2), b) päťuholník (5, 2, 0, 1), c) Petersenov graf (10, 3, 0, 1)

2.2 Základné vlastnosti silne regulárnych grafov

V tejto časti sú uvedené základné vlastnosti silne regulárnych grafov: komplement, vzťahy medzi parametrami a dôkaz spektrálnej vlastnosti silne regulárnych grafov, tzv. *celočíselné kritérium*.

2.2.1 Komplement

Zaujímavou vlastnosťou silne regulárnych grafov je, že ich komplementy sú tiež silne regulárne grafy.

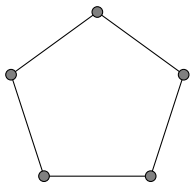
Veta 2.2. *Nech G je silne regulárny graf s parametrami (n, k, λ, μ). Potom komplentárny graf $\overline{G}$ (komplement) ku grafu G je tiež silne regulárny graf s parametrami (n, $\overline{k}$, $\overline{\lambda}$, $\overline{\mu}$), kde:*

$$\overline{k} = n - k - 1$$

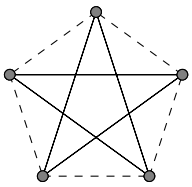
$$\overline{\lambda} = n - 2 - 2k + \mu$$

$$\overline{\mu} = n - 2k + \lambda$$

Príklad 2.1. *Pre silne regulárny graf pentagon s parametrami (5, 2, 0, 1) je komplement grafu silne regulárny graf s rovnakými parametrami (5, 2, 0, 1).*



Päťuholník (5, 2, 0, 1)



Komplement grafu (5, 2, 0, 1)

Silne regulárny graf G nazývame *primitívnym (primitive)*, ak on a aj jeho komplement sú súvislé. V opačnom prípade nazývame silne regulárny graf G *neprimitívnym (imprimitive)*. Nasledujúca lema dokazuje, že existuje iba jedna trieda neprimitívnych silne regulárnych grafov.

Lema 2.3. *Nech G je silne regulárny graf s parametrami (n, k, λ, μ). Potom nasledujúce tvrdenia sú ekvivalentné:*

- *graf G nie je súvislý graf*

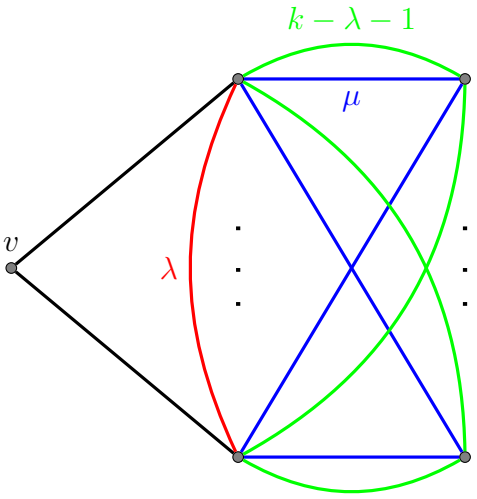
- $\mu = 0$
- $\lambda = k - 1$
- *graf G je izomorfný s grafom mK_{k+1} (m komponentový graf, kde každý komponent je kompletný graf s $k+1$ vrcholmi), pre nejaké $m > 1$*

Ďalej v našej práci sa zaoberáme iba primitívnymi silne regulárnymi grafmi, pre ktoré platia ďalšie vlastnosti.

2.2.2 Závislosť parametrov

Ďalšou vlastnosťou silne regulárneho grafu je závislosť jeho parametrov medzi sebou.

Veta 2.4. *Nech graf je silne regulárny graf s parametrami $G = (n, k, \lambda, \mu)$. Ak $\mu \neq 0$, potom $k(k-\lambda-1) = (n-k-1)\mu$, resp. $n = 1 + k + \frac{k(k-1-\lambda)}{\mu}$*



Obrázok 2.3: Zakreslenie silne regulárneho grafu, kde sú vrcholy rozdelené do troch skupín podľa vzdialenosti od vrchola v.

2.2.3 Integrálne kritérium

Základná spektrálna vlastnosť silne regulárnych grafov hovorí o vlastných číslach a ich násobnostiach.

Veta 2.5 (Integrálne kritérium)**.** *Ak existuje silne regulárny graf s parametrami $G = (n, k, \lambda, \mu)$, potom $\frac{2k+(n-1)(\lambda-\mu)}{\sqrt{(\lambda-\mu)^2+4(k-\mu)}}$ je celé číslo s rovnakou paritou ako $n-1$.*

Dôkaz. Nech matica A je matica susednosti grafu G. Z definície silne regulárneho grafu a rovnice (1.1) vieme, že matica A má vlastné číslo k s násobnosťou 1. Ukážeme, že matica A má iba 2 ďalšie vlastné čísla.

Z lemy 1.1 vyplýva, že koeficienty matice A^2 sú počty sledov dĺžky dva medzi príslušnými vrcholmi. Preto koeficienty v A^2 sú k, ak vrcholy sú rovnaké, λ ak vrcholy sú susedné a μ ak vrcholy sú nesusedné. Keďže matica susednosti komplementu G je $J - I - A$, dostávame vzťah, v ktorom matica A^2 je lineárnou kombináciou matíc $J,\ I,\ A$:

$$A^2 = kI + \lambda A + \mu(J - I - A)$$

Po úprave dostaneme:

$$A^2 = (\lambda - \mu)A + (k - \mu)I + \mu J \tag{2.1}$$

V k-regulárnom súvislom grafe je k vlastné číslo s násobnosťou 1 a vlastným vektorom $\mathbf{1}$. Podľa vety o ortogonálnosti vlastných vektorov reálnej symetrickej matice 1.2 vyplýva, že ostatné vlastné vektory matice A musia byť ortogonálne s $\mathbf{1}$.

Pre každý vlastný vektor v s vlastným číslom, ozn. θ, ktorý je ortogonálny s $\mathbf{1}$, dostávame $\mu Jv = 0$ a teda platí:

$$A^2v = (\lambda - \mu)Av + (k - \mu)Iv$$

Potom pre každé vlastné číslo $\theta \neq k$ platí:

$$\theta^2 = (\lambda - \mu)\theta + k - \mu$$

Dostali sme kvadratickú rovnicu, ktorá má 2 reálne riešenia (z vety o reálnosti vlastných čísel symetrickej matice 1.2), ozn. r a s:

$$r, s = \frac{\lambda - \mu \pm \sqrt{(\lambda - \mu)^2 + 4(k - \mu)}}{2}$$

Väčšinou predpokladáme, že $r > 0$ a $s \leq 0$. V primitívnych silne regulárnych grafoch vlastné číslo nemôže byť nulové a teda s je záporné číslo. Ako môžeme vidieť tieto 2 vlastné čísla sú závislé na parametroch silne regulárneho grafu.

Ďalej určíme násobnosti vlastných čísel r a s. Násobnosť vlastného čísla r budeme označovať f a násobnosť vlastného čísla s budeme označovať g.

Z vety o báze z vlastných vektorov 1.2 a rovnosti (1.2) vyplývajú 2 rovnice:

$$f + g + 1 = n$$

$$rf + sg + 1k = 0$$

Po úprave:

$$f = -\frac{(n-1)\,s + k}{r - s} \qquad g = \frac{(n-1)\,r + k}{r - s}$$

Po dosadení hodnôt r a s dostaneme:

$$g = \frac{1}{2}\left((n-1) + \frac{2k + (n-1)\,(\lambda - \mu)}{\sqrt{(\lambda - \mu)^2 + 4(k - \mu)}}\right)$$

Násobnosti vlastných čísel sú tiež závislé na parametroch silne regulárneho grafu. Sú to prirodzené čísla, z čoho vyplýva, že druhý výraz v zátvorke je tiež celé číslo s paritou ako $n - 1$. □

Príklad 2.2. *Pre Petersenov graf, ktorý má parametre (10, 3, 0, 1) sú vlastné čísla a ich násobnosti:*

$$r, s = \frac{-1 \pm \sqrt{9}}{2} = 1, -2 \qquad f, g = \frac{1}{2}\left(9 \pm \frac{-3}{\sqrt{9}}\right) = 5, 4$$

Petersenov graf spĺňa integrálne kritérium. Jeho maticové spektrum je: $(-2)^4, 1^5, 3$.

Príklad 2.3. *Pre päťuholník, ktorý má parametre (5, 2, 0, 1) sú vlastné čísla a ich násobnosti:*

$$r, s = \frac{-1 \pm \sqrt{5}}{2} \doteq 0.618, -1.618 \qquad f, g = \frac{1}{2}\left(4 \pm \frac{0}{\sqrt{5}}\right) = 2, 2$$

Päťuholník spĺňa integrálne kritérium. Jeho maticové spektrum je: $(0.618)^2, (-1.618)^2, 2$.

2.3 Typy silne regulárnych grafov

Na základe vlastností silne regulárnych grafov môžeme vyjadriť vzťah medzi f a g, pomocou ktorého rozdelíme grafy do dvoch typov:

$$g - f = \frac{2k + (n-1)\,(\lambda - \mu)}{\sqrt{(\lambda - \mu)^2 + 4(k - \mu)}}$$

- *Konferenčné (conference) grafy* sú také silne regulárne grafy s parametrami (n, k, λ, μ), pre ktoré platí: $f = g$. Z toho vyplýva, že:

  $$2k + (n-1)\,(\lambda - \mu) = 0$$

  Sú to tiež také silne regulárne grafy, ktoré majú rovnaké parametre ako ich komplementy. Z vlastností 2.4 a 2.2 vyplýva, že sú to silne regulárne grafy s parametrami $\left(n, \frac{(n-1)}{2}, \frac{(n-5)}{4}, \frac{(n-1)}{4}\right)$.

Nekonečná trieda konferenčných grafov sú napríklad *Paleyho grafy* $P(q)$, kde množina vrcholov je konečné pole $GF(q)$, kde q je prvočíselná mocnina (prime power) kongruetná s *1 mod 4*. Vrcholy u a v sú susedné práve vtedy, keď ak $u - v$ je nenulový štvorec v $GF(q)$ [Pal33].

Konferenčný graf je prepojený so tzv. *symetrickou konferenčnou maticou (conference matrix)* a preto veľkosť grafu musí byť $1 \pmod 4$ a zároveň súčet dvoch štvorcov [Pal33].

Vlastné čísla konferenčnej matice nemusia byť celé čísla narozdiel od druhého typu silne regulárnych grafov. Vlastné čísla r a s s násobnosťami f a g konferenčných grafov majú tvar:

$$r, s = \frac{-1 \pm \sqrt{n}}{2} \qquad f, g = \frac{n-1}{2}$$

Konferenčné grafy sú úplne známe pre malé $n < 30$, konkr. $n = 5, 9, 13, 17, 25, 29$. Pre $n = 21$ a 33 je dokázané, že neexistujú. Pre $n = 37$ je dokázaná existencia takýchto grafov, ale ich počet je neznámy [Srg15].

- Druhú skupinu tvoria všetky ostatné (primitívne) silne regulárne grafy, pre ktoré platí $f \neq g$. Z toho vyplýva, že $2k + (n-1)(\lambda - \mu) \neq 0$ a zároveň $\sqrt{(\lambda - \mu)^2 + 4(k - \mu)}$ musí byť štvorec, t.j. druhá mocnina nejakého celého čísla. Kvocient týchto dvoch vzťahov musí byť kongruentný s $n - 1 \bmod 2$. Z týchto vlastností vyplýva, že vlastné čísla r a s sú celé čísla.

Kapitola 3

Konštruktívna enumerácia kombinatorických objektov

V tejto kapitole sa budeme zaoberať generovaniu kombinatorických štruktúr, hlavne grafov, resp. matice susednosti. Cieľom konštruktívnej enumerácie kombinatorických objektov je nájsť všetky objekty s rovnakými vlastnosťami, ktoré nie sú izomorfné medzi sebou. Hlavnou myšlienkou je, že pre každú triedu objektov vygenerujeme jedného reprezentanta, pričom zahadzujeme novo vygenerované objekty, ktoré sú izomorfné s ním. V prvej časti tejto kapitoly zadefinujeme formálne *sadu nástrojov (framework)*, ktorý budeme ďalej v práci používať. V druhej časti ukážeme využitie izomorfizmu pri hľadaní objektov v našej sade nástrojov. Táto kapitola vychádza z dizertačnej práce od *Degraera* [Deg07].

3.1 Definícia

Kombinatorické štruktúry sú konečné a diskrétne štruktúry, z toho vyplýva, že počet možností pre hľadanie všetkých takýchto objektov s danou veľkosťou je tiež konečný a vieme ich hľadať systematicky. Najjednoduchšou metódou pre systematické hľadanie objektov je *spätné vyhľadávanie (backtracking)*. Predtým ako ukážeme, ako vieme hľadať objekty spätným prehľadávaním, zadefinujeme formálne klasifikáciu objektov, ktorú neskôr budeme používať. Vychádzame z knihy od *Dechterovej* [Dec03].

Definícia 3.1. *Systém s obmedzeniami* (Constraint system) $\mathcal{R} = (X, D, C)$ *pozostáva z konečnej množiny* $X = \{x_1, \ldots, x_n\}$*, premenných neprázdnej množiny domén* $D = (D_1, \ldots, D_n)$*, a množiny obmedzení* $C = (C_1, \ldots, C_n)$.

Doména D_i obsahuje všetky možné hodnoty prislúchajúcej premennej x_i. Obmedzenie C_j zahŕňa určitú podmnožinu premenných a špecifikuje dovolené kombinácie hodnôt príslušnej podmnožiny. Ak premennej je priradená hodnota z domény, potom je premenná *inštanciovaná*, v opačnom prípade je *neinštanciovaná*. Neinštanciované premenné budeme pre jednoduchosť označovať symbolom '?'. Inštancia množiny premenných $x_{i1}, \ldots, x_{ik}$ je k-tica usporiadaných párov $(\langle x_{i1}, d_{i1}\rangle, \ldots, \langle x_{ik}, d_{ik}\rangle)$, kde každá usporiadaná dvojica $\langle x, d\rangle$ označuje priradenie hodnoty d premennej x, kde d je z domény premennej x. Ďalej v práci budeme používať skrátený tvar tejto inštancie a to $(d_{i1}, \ldots, d_{ik})$, ak podmnožina premenných bude zrejmá z kontextu. *Riešenie* systému s obmedzeniami je úplná inštancia všetkých premenných, ktoré spĺňajú všetky obmedzenia. Čiastočná inštancia systému s obmedzeniami je *prípustná (feasible)*, ak spĺňa všetky obmedzenia, ktoré neobsahujú neinštanciované premenné. Je zrejmé, ak čiastočná inštancia $(d_1, \ldots, d_k)$ je prípustná, potom všetky čiastočné inštancie $(d_1, \ldots, d_j)$, kde $j < k$ sú prípustné. Samozrejme, ak čiastočná inštancia $(d_1, \ldots, d_k)$ nie je prípustná, potom všetky rozšírenia tejto inštancie $(d_1, \ldots, d_l)$, kde $l > k$ nie sú prípustné, resp. neexistuje riešenie pre $(d_1, \ldots, d_k)$.

Príklad 3.1. *Graf na 4 vrcholoch vieme zapísať do matice susednosti veľkosťou* 4×4*. Graf je jednoduchý a neorientovaný, to znamená že neobsahuje viacnásobné hrany a slučky. Potom matica susednosti je symetrická a diagonále má nuly a vieme ju zapísať nasledovne:*

$$A = \begin{pmatrix} 0 & x_1 & x_2 & x_3 \\ x_1 & 0 & x_4 & x_5 \\ x_2 & x_4 & 0 & x_6 \\ x_3 & x_5 & x_6 & 0 \end{pmatrix}$$

Pre regulárny graf rádu 4 so stupňom 2 je systém s obmedzeniami $\mathcal{R} = (X, D, C)$*, kde* $X = \{x_1, \ldots, x_6\}$ *s doménami* $D = \{D_1, \ldots, D_6\}$*, pričom pre každú doménu platí* $D_i = \{0, 1\}$*. Množinu obmedzení vieme ľahko odvodiť použitím matice susednosti* A*. Čiastočná inštancia, resp. úplná, je prípustná, ak súčet každého riadka alebo stĺpca je najviac, resp. práve 2. Inak povedané musí inštancia (čiastočná alebo úplná) spĺňať tieto lineárne rovnice:* $x_1 + x_2 + x_3 = 2$*,* $x_1 + x_4 + x_5 = 2$*,* $x_2 + x_4 + x_6 = 2$ *a* $x_3 + x_5 + x_6 = 2$*. Riešenie systému* $\mathcal{R}$ *zodpovedá matici susednosti* A *grafu, ktorý je rádu 4 a stupňom regulárnosti 2.*

Spätné vyhľadávanie (Backtracking) je rekurzívny algoritmus, v ktorom sa každá čiastočná inštancia premenných $X = \{x_1, \ldots, x_n\}$ zo systému $\mathcal{R} = (X, D, C)$ zachováva. Algoritmus začína so všetkými n premennými ako neinštanciovanými. V každom kroku v rekurzii

sa vyberie jedna určitá premenná, ktorá sa inštanciuje a systematicky sa priradia hodnoty z jej domény. Pre každú takúto hodnotu sa testuje prípustnosť čiastočnej inštancie premenných a to tak, že sa testujú iba tie obmedzenia, ktoré majú všetky premenné inštanciované. Ak je čiastočná inštancia premenných prípustná, potom sa rekurzívne zavolá, v opačnom prípade sa rekurzia nevykoná, pretože žiadne rozšírenie čiastočnej inštancie nemôže viesť k riešeniu daného systému. Ak všetky hodnoty premennej dávajú neprípustnú inštanciu, potom sa nachádzame v *mŕtvom bode (dead-end)*. Ak sa vyskúšajú všetky hodnoty premennej, potom sa algoritmus vráti *(backtrack)* späť ku vyvolanej procedúre. Uvádzame všeobecný pseudokód algoritmu 1, ktorý používa základnú stratégiu pri inštanciovaní premenných od najmenšieho indexu.

Algoritmus 1 Všeobecný algoritmus so spätným prehľadávaním pre systém obmedzení $\mathcal{R} = (X, D, C)$ s n premennými

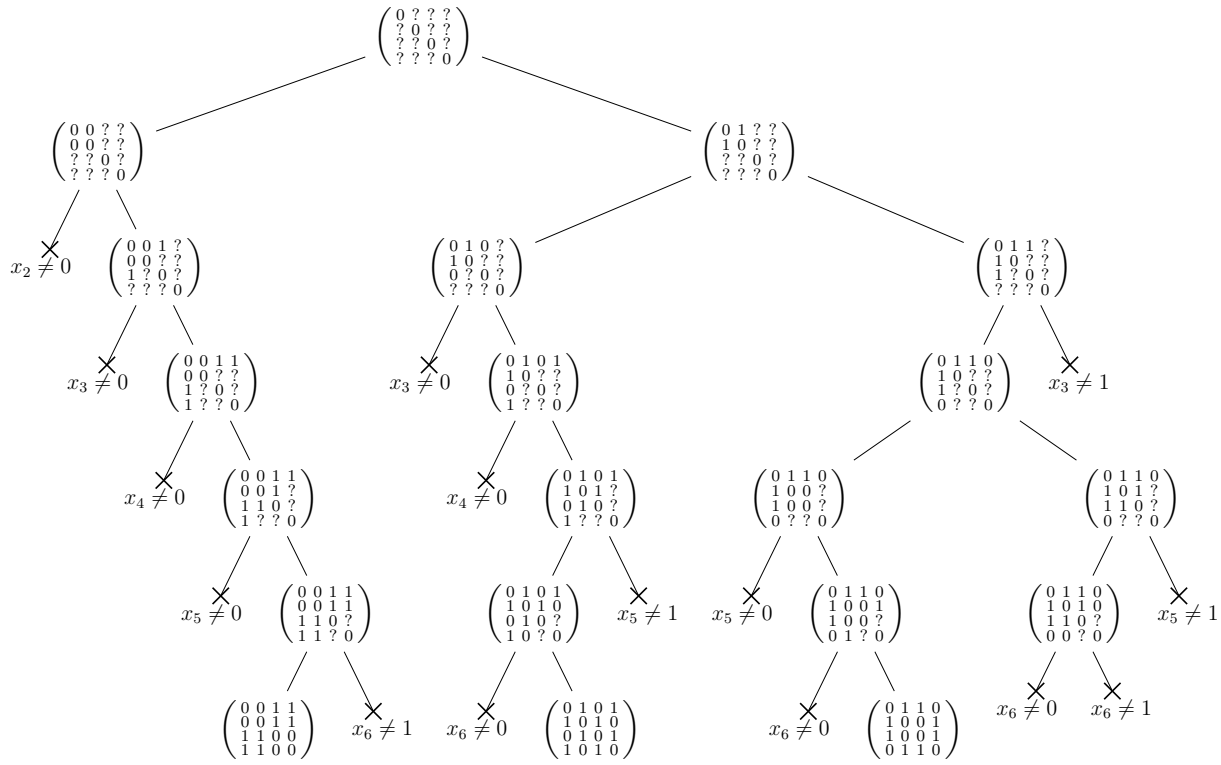
```
1: procedure SEARCH()
2:     BACKTRACK((), 0)

1: procedure BACKTRACK((d_1, ..., d_k): inštancia, k: číslo)
1:     if k = n then
2:         (d_1, ..., d_n) je riešenie systému R
3:     else
4:         for all d_{k+1} ∈ D_{k+1} do
5:             if (d_1, ..., d_{k+1}) je prípustná then
6:                 BACKTRACK((d_1, ..., d_{k+1}), k + 1)
```

Ľahko môžeme znázorniť *spätné prehľadávanie* pomocou prehľadávania stromu do hĺbky alebo rekurzívneho stromu. Vrcholy takéhoto stromu sú čiastočné prípustné inštancie a hrany reprezentujú rozšírenia z jednej inštancie na ďalšiu inštanciu. Ak čiastočná inštancia nie je prípustná, potom celý podstrom, ktorého koreň je daná inštancia, môžeme zahodiť *(prune)*. Listy stromu, vrcholy ktoré sú hĺbky n, sú riešenia systému s obmedzeniami $\mathcal{R} = (X, D, C)$. Množinou všetkých uzlov v strome budeme označovať $\mathcal{N}_{\mathcal{R}}$.

Príklad 3.2. *Pre systém s obmedzeniami $\mathcal{R}$, ktorý sme definovali v príklade 3.1, bude rekurzívny strom vyzerať ako na obrázku 3.1. V koreni stromu sú všetky premenné neinštanciované. Základná stratégia pri inštanciovaní premenných začína od najmenšieho indexu, to znamená že si vyberáme prvú neinštanciovanú premenná x_i, kde i je najmenší index. Ľavý syn, resp.*

pravý syn stromu zodpovedá rozšíreniu čiastočného prípustného systému s danou premennou hodnotou z domény 0, resp. 1. Vetva, ktorá je prešKrtnutá, znamená že dané rozšírenie na čiastočnú inštanciu nie je prípustné a preto môžeme celý podstrom zahodiť. Všimnime si, že náš strom nezodpovedá podmienkam C, tak ako sú uvedené v príklade 3.1. V tomto príklade využívame ďalšie obmedzenia, ktoré sú dôsledkami pôvodných. Napríklad musí platiť: $x_1 + x_2 \geq 1$, $x_2 + x_3 \geq 1$, $x_2 + x_4 \geq 1$, $x_3 + x_5 \geq 1$, *atď.*



Obrázok 3.1: Rekurzívny strom pre regulárny graf rádu 4 so stupňom 2

Systém s obmedzeniami spolu s algoritmom so spätným prehľadávaním nám poskytujú všeobecný návod na klasifikáciu kombinatorických štruktúr. Pri aplikácii na konkrétne kombinatorické štruktúry obtiažnosť klasifikácie spočíva v tom, ako najlepšie vieme prepísať matematické vlastnosti kombinatorických štruktúr do nášho systému týchto princípov. Sada nástrojov by mala minimalizovať počet možných výberov pri prechádzaní, mala by zahadzovať podstromy tak skoro ako je možné, minimalizovať prácu v každej volanej rekurzii a zároveň by mala byť čo najjednoduchšia kvôli detekcii možných chýb [KO05].

Ako si môžeme všimnúť, všeobecný algoritmus so spätným prehľadávaním nie je efektívny v zahadzovaní a to kvôli opakovanému prechádzaniu tých istých čiastočných prípustných alebo neprípustných inštancií počas prehľadávania. To vieme vylepšiť dynamickým inštanci-

ovaním premenných rôznymi stratégiami. Stratégie môžu dynamicky inštanciovať premenné buď od najmenšieho indexu premennej po najväčší alebo naopak.

Jednou zo stratégií je tzv. *pozeranie dopredu (look ahead)*, ktorý počas priradenia hodnoty z domény do premennej zistí, či hodnoty z domén, ktoré nie sú inštanciované nie sú v konflikte s čiastočnou inštanciou, t. j. rozšírenie čiastočnej inštancie o hodnotu z domén by bolo neprípustné. Ak áno, potom hodnoty z domén môžeme odstrániť. Táto stratégia si vyžaduje navyše zložitosť pre každú inštanciu premennej, ale mŕtve body v strome nájde najskôr ako je možné a väčšinou prechádza iba malú časť rekurzívneho stromu.

Ďalšou zo stratégii je *dynamické usporiadanie premenných (dynamic variable ordering)*, ktorá vždy keď si vyberá novú premennú na inštanciovanie, dynamicky sa rozhodne počas hľadania, ktorá to bude podľa nejakej heuristiky. Základná heuristika je výber premennej s najmenším počtom hodnôt v doméne v momente hľadania. Táto stratégia nájde mŕtve body najskôr ako je možné (doména nejakej neinštanciovanej premennej je prázdna množina).

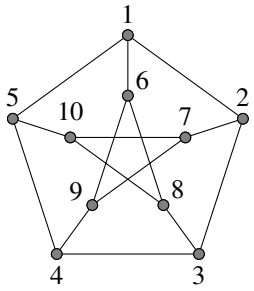
Poslednú stratégiu, ktorú v práci spomíname je *systém s pamäťou (constraint recording)*. Táto stratégia inštanciuje premenné od najväčšieho po najmenší index a počas hľadania ak stretne s mŕtvym koncom, zapamätá si stav a tak sa v budúcnosti bude vyhýbať rovnakým konfliktom.

Podrobné informácie o týchto a ďalších stratégii je možné nájsť v [HE79], [Kum92] a [Dec03].

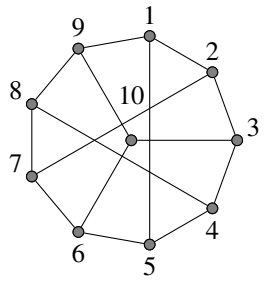
3.2 Využitie izomorfizmu

V tejto časti ukážeme využitie teórie grúp v našej sade nástrojov. Budeme sa odvolávať na definície a vety, ktoré sú uvedené v 1.3.

Príklad 3.3. *Nech G a H sú grafy, ktoré sú na obrázku 3.2. Ľahko overíme, že permutácia $\pi = (6\,9\,8\,10) \in Sym(10)$ je izomorfizmus medzi G a H, t. j. $\pi \cdot G = H$.*



G



H

Na overenie dvoch objektov $X, Y \in \Delta$ na izomorfizmus musíme nájsť izomorfizmus z X do Y alebo naopak. V praxi zvyčajne na overenie $X \cong_G Y$ porovnávame kanonických reprezentantov X a Y pomocou kanonického reprezentatívneho zobrazenia.

Definícia 3.2. *Kanonické reprezentatívne zobrazenie pre akciu grupy G na Δ je funkcia $\phi : \Delta \to \Delta$ taká, že $\phi(X) \cong_G X$ pre každé $X \in \Delta$ a zároveň ak $X \cong_G Y$, potom $\phi(X) = \phi(Y)$ pre každé $X, Y \in \Delta$.*

Pre dané kanonické reprezentatívne zobrazenie ϕ je objekt $X \in \Delta$ v *kanonickom tvare*, ak $\phi(X) = X$. Zároveň $\phi(X)$ je *kanonický reprezentant* GX, orbita X v G. Z toho vyplýva, že $X \cong_G Y$ práve vtedy, keď $\phi(X) = \phi(Y)$. To znamená, že testovanie dvoch objektov X a Y na izomorfizmus v podstate porovnáva ich kanonických reprezentantov.

Definícia 3.3. *Invariant pre akciu grupy G na Δ je funkcia $\xi : \Delta \to \Delta$ taká, že $\xi(X) = \xi(Y)$, ak $X \cong_G Y$ pre každé $X, Y \in \Delta$.*

Definícia 3.4. *Certifikát je taký invariant ξ pre akciu grupy G na Δ, že $\xi(X) = \xi(Y)$ práve vtedy, keď $X \cong_G Y$ pre každé $X, Y \in \Delta$.*

Invariant je vlastnosť objektov z Δ, ktorá sa nemení po vykonaní akcie grupy G na Δ. Pre invariant platí ak $\xi(X) \neq \xi(Y)$, potom $X \ncong_G Y$, ale zároveň opačná implikácia nemusí platiť, t.j. ak $X \ncong_G Y$, potom $\xi(X) \neq \xi(Y)$. Ak platia obidve implikácie, potom daný invariant je certifikát.

3.3 Izomorfné zahadzovanie

Hlavným cieľom konštruktívnej enumerácie objektov je hľadanie a eliminácia izomorfných objektov. Vo všeobecnosti, nech Δ reprezentuje množinu všetkých kombinatorických objektov, na ktorej vykonávame akciu grupy G, potom hlavným cieľom izomorfného zahadzovania

je vo všeobecnosti vygenerovať práve jedného reprezentanta pre každú orbitu z $G \parallel \Delta$. Obtiažnosť odstraňovania izomorfných kópií počas prechádzania rekurzívneho stromu je úmerná počtu izomorfných objektov. Dokonca už pre malé kombinatorické objekty by generovanie bolo neefektívne alebo nemožné. Okrem toho sa dajú využiť rôzne techniky izomorfného zahadzovania, ktoré sa počas prechádzania vyhýbajú vetvám v rekurzívnom strome, ktoré sú rovnaké, t.j. obsahujú listy, ktoré nezodpovedajú objektom v Δ.

McKay v [MW91] rozdeľuje algoritmy pre hľadanie neizomorfných objektov pomocou konštruktívnej enumerácie do troch kategórii. Prvá kategória je *usporiadané generovanie (orderly generation)*, kde sa každej orbite priradí práve jeden kanonický objekt. Tieto algoritmy boli nezávisle prvýkrát predstavené *Readom* a *Faradževom*. Využitie tohto prístupu môžeme nájsť pri klasifikácii rôznych kombinatorických objektov, napríklad *kubických grafov* [Bri96] alebo *regulárnych grafov* [Mer99]. Druhou kategóriou je *kanonické rozšírenie (canonical augmentation)*. Tento prístup predstavil *McKay* v [MW91]. Zaoberať sa budeme jeho jednoduchšou verziou a to *slabé kanonické rozšírenie (weak canonical augmentation)* [KO05]. V tomto prístupe sú objekty konštruované kanonickým spôsobom namiesto toho, aby museli byť v kanonickom tvare. Počas prehľadávania systematicky zahadzujeme izomorfné objekty, ktoré sú príbuzné. Využitie tohto prístupu môžeme nájsť pri klasifikácii rôznych kombinatorických objektov, napríklad *Steinerov systém trojíc* [KO04] alebo *latinské štvorce* [MMM07]. Poslednou kategóriou je *princíp homomorfizmu (homomorphism principle)*, kde sa využívajú ďalšie grupové výpočty, ktoré popisujú vzťah medzi grupou automorfizmov a objektmi. Tento prístup predstavili *Laue, Grüner a Meringer*. Využitie môžeme nájsť pri klasifikácii rôznych kombinatorických objektov, napríklad *molekulárnych grafov* [GKL96] alebo *t-blokových plánov* [Sch93].

Pre našu triedu objektov využijeme algoritmus z prvej kategórie a to usporiadané generovanie. Tento algoritmus vieme popísať našou sadou nástrojov, čiže cez systém s obmedzeniami a rekurzívnym stromom. Predpokladajme systém s obmedzeniami $\mathcal{R} = (X, D, C)$ s n premennými, kde konečná množina listov nám zodpovedá množine Δ, na ktorej vykonávame akciu grupy G. Na popis algoritmu usporiadaného generovania rozšírime akciu grupy G na $\mathcal{N}_\mathcal{R}$, množinu všetkých uzlov v rekurzívnom strome.

Predpokladajme kanonické reprezentatívne zobrazenie ϕ pre akciu grupy G na $\mathcal{N}_\mathcal{R}$. V usporiadanom generovaní budeme zahadzovať všetky uzly v rekurzívnom strome, ktoré nie sú v kanonickom tvare vzhľadom na ϕ. Pri postupnom generovaní práve jedného reprezentanta

pre každú orbitu z $G \parallel \Delta$ musí pre kanonické reprezentatívne zobrazenie ϕ platiť, že $\phi(X) \in \mathcal{N}_\mathcal{R}$ a ak $\phi(Y) = Y$, potom $\phi(Z) = Z$ pre každé $Y, Z \in \mathcal{N}_\mathcal{R}$ tak, že uzol Z je rodičom uzla Y.

Z toho vyplýva, že len jeden reprezentant pre každú orbitu z $G \parallel \Delta$ bude vygenerovaný, lebo len kanonický reprezentant môže byť riešením pre systém s obmedzeniami $\mathcal{R}$. Taktiež pre každé $X \in \Delta$ máme všetky uzly na ceste z koreňa do listu $\phi(X)$ v kanonickom tvare.

Uvádzame všeobecný pseudokód algoritmu pre usporiadané generovanie 2, ktorý tiež používa základnú stratégiu pri inštanciovaní premenných od najmenšieho indexu ako algoritmus 1.

Algoritmus 2 Všeobecný algoritmus so spätným prehľadávaním usporiadaného generovania pre systém obmedzení $\mathcal{R} = (X, D, C)$ s n premennými, ktorý využíva kanonické reprezentatívne zobrazenie ϕ s akciou grupy G na $\mathcal{N}_\mathcal{R}$

```
1: procedure SEARCH()
2:     BACKTRACK((), 0)

1: procedure BACKTRACK((d_1, ..., d_k): inštancia, k: číslo)
1:     if (d_1, ..., d_k) je v kanonickom tvare then
2:         if k = n then
3:             (d_1, ..., d_n) je riešenie systému R
4:         else
5:             for all d_{k+1} ∈ D_{k+1} do
6:                 if (d_1, ..., d_{k+1}) je prípustná then
7:                     BACKTRACK((d_1, ..., d_{k+1}), k + 1)
```

Príklad 3.4. *V tomto príklade využijeme predchádzajúci systém s obmedzeniami $\mathcal{R} = (X, D, C)$ z príkladu 3.1, ktorý popisuje postup pri hľadaní všetkých regulárnych grafov stupňa 4 s rádom 2, ozn. Δ. Akcia grupy $Sym(4)$ na množine Δ a zároveň na množine $\mathcal{N}_\mathcal{R}$ pozostavajúcej z matíc veľkosti 4×4 zodpovedá rekurzívnemu stromu, t.j. akcia grupy $Sym(4)$ na $\mathcal{N}_\mathcal{R}$ je definovaná $(\pi \cdot M)_{x,y} = (M)_{x^\pi, y^\pi}$, kde pre každé $M \in \mathcal{N}_\mathcal{R}$, $\pi \in Sym(4)$ a $x, y \in \{1, \ldots, 4\}$. Z toho vyplýva, že dve matice sú izomorfné práve vtedy, keď vieme dostať z jednej druhú pomocou permutácie riadkov a stĺpcov naraz.*

Definujeme usporiadanie $? < 0 < 1$ a pre každé $M \in \mathcal{N}_\mathcal{R}$ je 6-tica nad $?, 0, 1$, ktorá zodpovedá hodnotám premenných $x_1, \ldots, x_6$. Základná stratégia nám ďalej definuje ako 6-

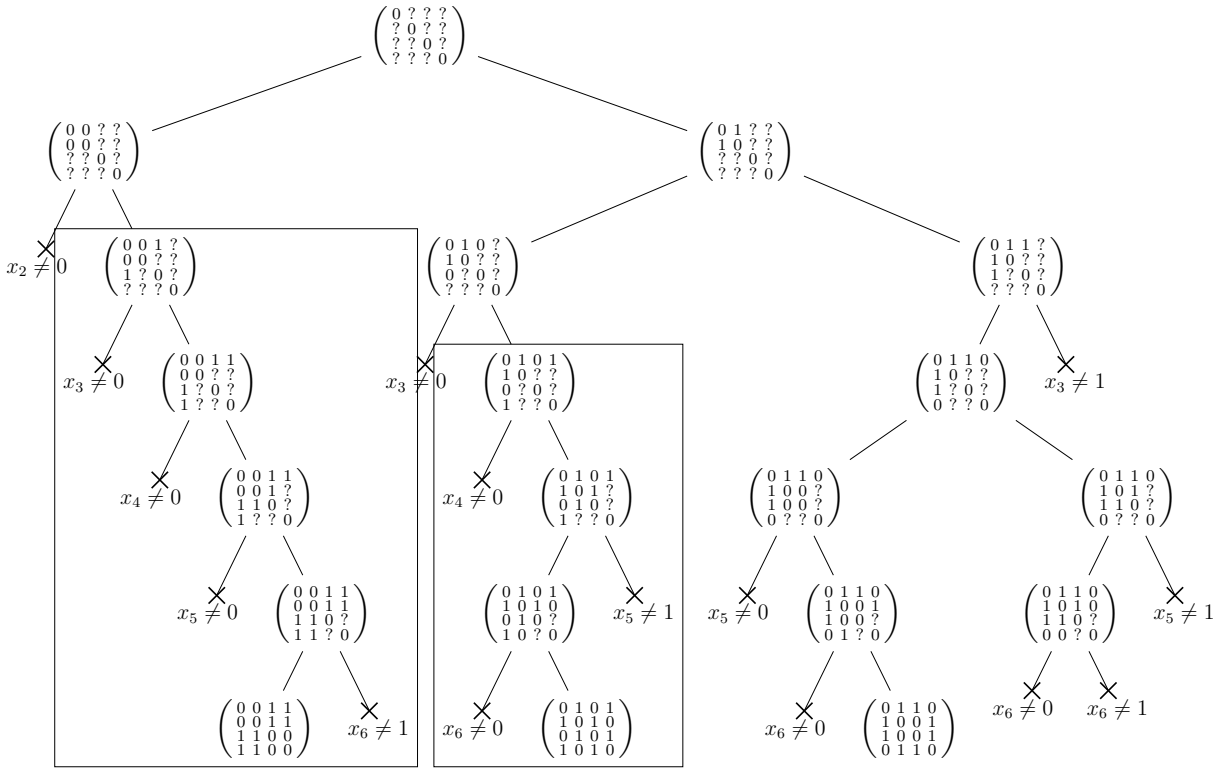
tica v každom uzle v rekurzívnom strome sa vyskytuje v lexikografickom usporiadaní. Matica $M \in \mathcal{N}_{\mathcal{R}}$ *je v kanonickom tvare, keď 6-tica každej matice orbity* M *vzhľadom na* $Sym(4)$ *je lexikograficky menšia alebo rovná 6-tici matice* M.

Na obrázku 3.3 môžeme vidieť rekurzívny strom pre algoritmus 2. Môžeme si všimnúť, že pre každý uzol, ktorý je v kanonickom tvare, je aj jeho rodič v kanonickom tvare. Zvýraznenou oblasťou sú označené uzly, ktoré algoritmus vynechal pretože nie sú v kanonickom tvare a zároveň aj ich deti. Konkrétne prvý zvýraznený podstrom môžeme zahodiť pretože:

$$\pi \cdot \begin{pmatrix} 0&0&1&?\\ 0&0&?&?\\ 1&?&0&?\\ ?&?&?&0 \end{pmatrix} = \begin{pmatrix} 0&1&0&?\\ 1&0&?&?\\ 0&?&0&?\\ ?&?&?&0 \end{pmatrix}, kde\ \pi = (2\,3) \in Sym(4)$$

Z toho dostávame, že $(0,1,?,?,?,?) < (1,0,?,?,?,?)$. *Podobne pre druhý zvýraznený podstrom dostaneme* $(1,0,1,?,?,?) < (1,1,0,?,?,?)$ *a môžeme ho zahodiť pretože:*

$$\pi \cdot \begin{pmatrix} 0&1&0&1\\ 1&0&?&?\\ 0&?&0&?\\ 1&?&?&0 \end{pmatrix} = \begin{pmatrix} 0&1&1&0\\ 1&0&?&?\\ 1&?&0&?\\ 0&?&?&0 \end{pmatrix}, kde\ \pi = (3\,4) \in Sym(4)$$



Obrázok 3.3: Rekurzívny strom pre regulárny graf rádu 4 so stupňom 2 bez izomorfných grafov

Kapitola 4

Orbitné matice

Veľkosť matice susednosti pre silne regulárne grafy, ktoré skúmame je veľká a preto hlavnou motiváciou je jej redukcia. Jednou z možností je využitie automorfizmu grúp rádu p, kde p je prvočíslo. V tejto kapitole vychádzame hlavne z dizertačnej práce *Behbahaniho* [Beh09].

4.1 Automorfizmus silne regulárnych grafov rádu p

Nech G je silne regulárny graf s grupou automorfizmom rádu p, kde p je prvočíslo. Nech $Fix(x) = \{v \in G | v^x = v\}$ je množina fixných vrcholov grafu G a nech Ω tvorí indukovaný podgraf týchto vrcholov. Nech x je generátor tejto grupy. Potom pre každé u, v z Ω platí:

$$\begin{aligned} |Fix(x)| &\equiv n \mod p \\ d_\Omega(u) &\equiv k \mod p \\ |N_\Omega(u) \cap N_\Omega(v)| &\equiv \lambda \mod p \text{ ak } u \sim v \\ |N_\Omega(u) \cap N_\Omega(v)| &\equiv \mu \mod p \text{ ak } u \not\sim v \end{aligned}$$

4.2 Definícia

Nech G je silne regulárny graf s parametrami (n, k, λ, μ). Predpokladajme automorfizmus na G rádu p, kde množinu vrcholov G rozdelíme do b *orbít* $O_1, O_2, \ldots, O_b$. Potom nech $n_i = |O_i|$ pre $1 \leq i \leq b$.

Nech $v_1, v_2, \ldots, v_n$ je usporiadanie vrcholov grafu G, ktoré zachováva usporiadanie $(O_1, O_2, \ldots, O_b)$. Resp. ak $i < j$ potom pre všetky $v_l \in O_i$ a $v_m \in O_j$ platí $l < m$.

Podľa usporiadania vrcholov nám orbity delia maticu susednosti A grafu G na podmatice $A = [A_{ij}]$, kde A_{ij} je matica susednosti vrcholov orbity O_i oproti vrcholom orbity O_j.

Príklad 4.1. *Zoberme si malý silne regulárny (Petersenov) graf, ktorý má parametre (10, 3, 0, 1). Petersenov graf má automorfizmus rádu 3 s jedným fixným vrcholom, a preto vieme jeho maticu susednosti znázorniť nasledovne. Prvý riadok je fixný vrchol a ďalšie trojice riadkov zodpovedajú orbitám veľkosti 3.*

$$A = \left(\begin{array}{c|ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right)$$

Potom definujeme nové 3 matice $C = [c_{ij}]$, $R = [r_{ij}]$ a N typu $b \times b$ nasledovne:

$$c_{ij} = \text{súčet stĺpca v } A_{ij}$$

$$r_{ij} = \text{súčet riadka v } A_{ij}$$

$$N = \text{diag}(n_1, n_2, \ldots, n_b)$$

Maticu C, resp. R budeme nazývať *orbitnou maticou* grafu G, ktorá nám dáva informácie o štruktúre matice susednosti grafu. V orbitnej matici sú prvé riadky fixné vrcholy. Hodnota c_{ij} (r_{ij}) v orbitnej matici nezávisí od výberu riadku (stĺpca) matice A_{ij}.

Príklad 4.2. *Z príkladu 4.1 by to boli matice:*

$$C = \left(\begin{array}{c|ccc} 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 \\ 3 & 1 & 1 & 0 \end{array}\right) \qquad R = \left(\begin{array}{c|ccc} 0 & 0 & 0 & 3 \\ \hline 0 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{array}\right) \qquad N = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

4.3 Vlastnosti orbitných matíc

Z definície matíc R a C je zrejmé, že platí vzťah:

$$R = C^T$$

Maticu A^2 budeme označovať ako maticu W, kde W_{uv} počíta sledy $u - v$ dĺžky 2. Maticu W tiež vieme rozdeliť na bloky $[W_{ij}]$, kde i a j sú indexy orbít.

Príklad 4.3. *Z príkladu 4.1 matica W bude vyzerať takto:*

$$W = A^2 = \left(\begin{array}{ccc|ccc|ccc} 3 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 3 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 3 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 3 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 1 & 0 & 3 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 3 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 3 & 1 & 1 \\ \hline 0 & 1 & 0 & 1 & 0 & 1 & 1 & 3 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 3 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{array}\right)$$

Potom definujeme maticu S s veľkosťou $b \times b$, $S = [s_{ij}]$ takú, že:

$$s_{ij} = \text{súčet všetkých hodnôt v } W_{ij}$$

Hodnota s_{ij} nezávisí od výberu riadku (stĺpca) v W_{ij}. Z vlastností silne regulárnych grafov vyplýva vzťah (2.1) pomocou, ktorého vieme vyjadriť pre jednotlivé podmatice W_{ij} vzťah $W_{ij} = (\lambda - \mu)A_{ij} + \delta_{ij}(k - \mu)I + \mu J$, kde δ funkcia je *Kroneckerova delta funkcia* definovaná nasledovne: $\delta_{ij} = 1$, ak $i = j$ inak 0 a zároveň dimenzia matice I je $n_i \times n_i$ a matice J je $n_i \times n_j$. Z toho potom dostávame vzťah, ktorý definuje maticu S:

$$s_{ij} = (\lambda - \mu)c_{ij}n_j + \delta_{ij}(k - \mu)n_j + \mu n_i n_j \tag{4.1}$$

Príklad 4.4. *Matica S z príkladu 4.3 bude vyzerať nasledovne:*

$$S = \begin{pmatrix} 3 & 3 & 3 & 0 \\ 3 & 15 & 3 & 6 \\ 3 & 3 & 15 & 6 \\ 0 & 6 & 6 & 15 \end{pmatrix}$$

Zároveň sa dá ukázať, že platí nasledovná veta.

Veta 4.1. *Pre matice C, R, N a S platí:*

$$CNR = S$$

Dôkaz. Nech α_i je vektor veľkosti n definovaný takto:

$$\alpha_i(j) = \begin{cases} 1 & \text{ak } j \in O_i \\ 0 & \text{inak} \end{cases}$$

Potom nech vektory α_i a α_j sú zvolené tak, aby platil vzťah: $s_{ij} = \alpha_i W \alpha_j^T = \alpha_i A^2 \alpha_j^T$. Nech β_i pre $1 \leq i \leq b$ je vektor veľkosti n taký, že $\beta_i(j) = c_{ij}$ ak $j \in O_i$. A podobne nech γ_i pre $1 \leq i \leq b$ je vektor veľkosti n taký, že $\gamma_i(j) = r_{ij}$ ak $j \in O_i$. Potom platí:

$$s_{ij} = \alpha_i A^2 \alpha_j^T = (\alpha_i A)\left(\alpha_j^T\right) = \beta_i \gamma_j^T = \sum_{k=1}^{b} c_{ik} n_k r_{kj} = (CNR)_{ij}$$

□

Pomocou tejto vety vieme odvodiť sústavu rovníc, ktorej riešenie nám dáva všetky možné varianty pre riadky orbitnej matice C. Jedinú informáciu, ktorú potrebujeme na odvodenie rovníc sú parametre silne regulárneho grafu $G = (n, k, \lambda, \mu)$ a počet pevných bodov. Sústava rovníc je nezávislá od matice susednosti A grafu G. Z toho vyplýva, že pomocou riešenia rovníc vieme nájsť orbitnú maticu C bez znalosti matice A.

Matica susednosti silne regulárneho grafu G je symetrická, z toho vyplýva, že pre orbitnú maticu musí platiť pre $i < j$: $c_{ij} = c_{ji}\left(\frac{n_j}{n_i}\right)$. Z toho vyplýva, že stačí generovať iba hornú trojuholníkovú časť v orbitnej matici s diagonálou, ostatné prvky si vieme vypočítať.

Z Vety 4.1 vyplýva tento vzťah:

$$s_{ij} = \sum_{k=1}^{b} c_{ik} c_{jk} n_k \tag{4.2}$$

Na základe rovnice (4.1) a (4.2) vieme vytvoriť sústavu rovníc, ktoré orbitná matica musí spĺňať. Ak generujeme orbitnú maticu postupne po riadkoch dostaneme lineárnu sústavu rovníc, ktorú musí daný riadok spĺňať. Najprv v orbitnej matici generujeme riadky, ktoré majú orbitu veľkosti 1 a po nich nasledujú riadky, ktoré majú orbitu veľkosti p. Pomocou tohto vieme popísať hľadanie orbitných matíc pomocou našej sady nástrojov, ktoré sme definovali v predchádzajúcej kapitole. Máme systém s obmedzeniami $\mathcal{R} = (X, D, C)$, kde

prvky $X = \{c_{11}, c_{12}, \ldots, c_{1b}, c_{22}, c_{23}, \ldots, c_{2b}, \ldots, c_{bb}\}$ s doménami $D = \{D_1, D_2, D3\}$, kde $D_1 = \{0, 1\}$, $D_2 = \{0, p\}$ a $D_3 = \{0, \ldots, p\}$. Doména D_1 je pre premenné, ktoré sú na riadku s orbitou veľkosti 1. Pre premenné, ktoré sú na riadku s orbitou p majú buď doménu D_2, ak sa nachádzajú v stĺpci s orbitou veľkosti 1 alebo D_3, ak sa nachádzajú s orbitou veľkosti p. Množinu obmedzení C tvorí predchádzajúca sústava rovníc a zároveň ďalšie podmienky, ktoré vieme z nich odvodiť. Tie ukážeme v nasledujúcej časti.

Nech f je počet orbít veľkosti 1 a nech o je počet orbít veľkosti p. Potom musí platiť:

$$f = n - po \tag{4.3}$$

V tomto prípade máme dva možné typy riadkov (stĺpcov). *Fixné riadky (stĺpce)* sú tie riadky (stĺpce), ktoré majú orbitu veľkosti 1 a zároveň *orbitné riadky (sĺpce)* sú tie riadky (stĺpce), ktoré majú orbitu veľkosti p. Na začiatku nebudeme brať do úvahy usporiadanie hodnôt v riadkoch orbitnej matice, zaujíma nás ich rozdelenie. Rozdelenie hodnôt budeme nazývať *prototypom orbitnej matice*. Každý prototyp nám hovorí o všetkých možných počtoch každého čísla v každom type riadka orbitnej matice C.

Predpokladajme ľubovoľný fixný riadok v orbitnej matici C. Možné hodnoty pre tento riadok sú 0 alebo 1 bez ohľadu na to, či sme nad fixnými alebo nad orbitnými stĺpcami. Nech x_0 je počet núl v danom riadku nad fixnými stĺpcami a podobne nech x_1 je počet jednotiek. Analogicky nech y_0 je počet núl v danom riadku nad orbitnými stĺpcami a y_1 je počet jednotiek. Zároveň platí súčet hodnôt riadku matice A je rovný k, čiže $x_1 + py_1 = k$. Z toho dostávame sústavu lineárnych rovníc pre fixný riadok:

$$\begin{aligned} x_0 + x_1 &= f \\ y_0 + y_1 &= o \\ x_1 + py_1 &= k \end{aligned} \tag{4.4}$$

Prototyp pre fixný riadok budeme nazývať každé také riešenie systému lineárnych rovníc, pre ktoré platí $x_0,\ x_1,\ y_0,\ y_1 \geq 0$.

Predpokladajme ľubovoľný orbitný riadok v orbitnej matici C. Možné hodnoty pre tento riadok sú 0 alebo p nad fixnými stĺpcami a $0, 1, \ldots p$ nad orbitnými stĺpcami. Nech x_0, resp. x_p je počet núl, resp. prvkov p v danom riadku nad fixnými stĺpcami a nech y_i, kde $i = 0, \ldots, p$ je počet prvkov i nad orbitnými stĺpcami. Pre fixný riadok platia podobné rovnice ako pre orbitný riadok a zároveň z rovnice (4.2) dostávame: $p^2 x_p + \sum_{i=1}^{p} pi^2 y_i = s_{ii}$. Z toho

dostávame sústavu lineárnych rovníc pre orbitný riadok:

$$\begin{aligned} x_0 + x_p &= f \\ y_0 + y_1 + \ldots + y_p &= o \\ x_p + y_1 + \ldots + py_p &= k \\ px_p + y_1 + \ldots + p^2 y_p &= s_{ii}/p \end{aligned} \tag{4.5}$$

Hodnotu s_{ii} vieme vypočítať pomocou rovnice (4.1), kde každá hodnota c_{ii} vygeneruje novú sústavu rovníc. Potom platí nasledovná veta.

Veta 4.2. *Ak n_i je nepárne číslo, potom c_{ii} musí byť párne číslo.*

Dôkaz. Nech H je indukovaný podgraf orbitou O_j a nech jeho matica susednosti je A_{ii}. Graf H je regulárny so stupňom rádu c_{ii}. Z toho vyplýva, že počet hrán grafu H je rovný $(n_i c_{ii})/2$ a z toho dostávame vzťah:

$$2|E(H)| = n_i c_{ii}$$

Ľahko môžeme pozorovať, že ak n_i je nepárne číslo, potom c_{ii} musí byť párne číslo. □

Prototyp pre orbitný riadok budeme nazývať každé také riešenie systému lineárnych rovníc, pre ktoré platí $x_0,\ x_p,\ y_0,\ \ldots,\ y_p \geq 0$.

Z rovnice (4.3) vyplýva, že najmenšia hodnota pre f musí byť hodnota $a = n \mod p$. Všetky možné hodnoty pre f sú $a, a+p, \ldots, a + \lfloor \frac{n-a}{p} \rfloor$. Z toho vyplývajú nasledujúce tvrdenia.

Veta 4.3. *Ak existuje prototyp pre fixný riadok s f fixnými vrcholmi a $f \geq 2p$, potom existuje fixný prototyp s $f - p$ fixnými vrcholmi.*

Dôkaz. Nech existuje prototyp pre fixný riadok s f fixnými vrcholmi, t. j. riešenie lineárnej sústavy (4.4) nech je (x_0, x_1, y_0, y_1). Predpokladajme lineárnu sústavu pre fixný prototyp s $f - p$ fixnými vrcholmi:

$$\begin{aligned} \bar{x}_0 + \bar{x}_1 &= f - p \\ \bar{y}_0 + \bar{y}_1 &= \frac{n - (f-p)}{p} = o + 1 \\ \bar{x}_1 + p\bar{y}_1 &= k \end{aligned}$$

Ukážeme, že táto lineárna sústava má riešenie, ktorej premenné sú nezáporné a celé čísla. Z predpokladu $f \geq 2p$ vyplývajú dve možnosti, buď $x_0 \geq p$ alebo $x_1 \geq p$. Ak $x_0 \geq p$, potom

z lineárnej sústavy vyplýva $\bar{x}_0 = x_0 - p$, $\bar{x}_1 = x_1$, $\bar{y}_0 = y_0 + 1$ a $\bar{y}_1 = y_1$. Z toho vyplýva, že $\bar{x}_0,\ \bar{x}_1,\ \bar{y}_0,\ \bar{y}_1 \geq 0$. Ak $x_1 \geq p$, potom z lineárnej sústavy vyplýva $\bar{x}_0 = x_0$, $\bar{x}_1 = x_1 - p$, $\bar{y}_0 = y_0$ a $\bar{y}_1 = y_1 + 1$. Tiež z toho vyplýva, že $\bar{x}_0,\ \bar{x}_1,\ \bar{y}_0,\ \bar{y}_1 \geq 0$. □

Veta 4.4. *Ak existuje prototyp pre orbitný riadok s f fixnými vrcholmi a $f \geq 2p$, potom existuje orbitný prototyp s $f - p$ fixnými vrcholmi.*

Dôkaz. Nech existuje prototyp pre orbitný riadok s f fixnými vrcholmi, t. j. riešenie lineárneho systému (4.5) nech je $(x_0,\ x_p,\ y_0,\ \ldots,\ y_p)$. Predpokladajme lineárnu sústavu pre orbitný prototyp s $f - p$ fixnými vrcholmi:

$$\begin{aligned} \bar{x}_0 + \bar{x}_p &= f - p \\ \bar{y}_0 + \bar{y}_1 + \ldots + \bar{y}_p &= \frac{n - (f - p)}{p} = o + 1 \\ \bar{x}_p + \bar{y}_1 + \ldots + p\bar{y}_p &= k \\ p\bar{x}_p + \bar{y}_1 + \ldots + p^2\bar{y}_p &= s_{ii}/p \end{aligned}$$

Ukážeme, že táto lineárna sústava má riešenie, ktorej premenné sú nezáporné a celé čísla. Z predpokladu $f \geq 2p$ vyplývajú dve možnosti, buď $x_0 \geq p$ alebo $x_p \geq p$. Ak $x_0 \geq p$, potom z lineárnej sústavy vyplýva $\bar{x}_0 = x_0 - p$, $\bar{x}_p = x_p$, $\bar{y}_0 = y_0 + 1$ a $\bar{y}_i = y_i$ pre $1 \leq i \leq p$. Z toho vyplýva, že $\bar{x}_0,\ \bar{x}_p,\ \bar{y}_0, \ldots,\ \bar{y}_p \geq 0$. Ak $x_p \geq p$, potom z lineárnej sústavy vyplýva $\bar{x}_0 = x_0$, $\bar{x}_p = x_p - p$, $\bar{y}_i = y_i$ pre $0 \leq i \leq p - 1$ a $\bar{y}_p = y_p + 1$. Z toho vyplýva, že $\bar{x}_0,\ \bar{x}_p,\ \bar{y}_0, \ldots,\ \bar{y}_p \geq 0$. □

Z týchto viet vyplýva, že ak nenájdeme prototyp pre fixný (orbitný) riadok s väčším počtom pevných vrcholov, potom nemusíme brať do úvahy prototypy s väčšími f fixnými vrcholmi, kde $f \geq 2p$.

4.4 Odhad počtov fixných vrcholov

V tejto časti ukážeme ako pomocou vlastných čísel silne regulárneho grafu a ďalších tvrdení vieme zhora ohraničiť počet fixných vrcholov pre automorfizmus grupy rádu p.

Nasledujúca lema nám dáva obmedzenie na riadky matice susednosti indukovaného podgrafu grafu G, ktorý pozostáva z orbitných vrcholov.

Lema 4.5. *Nech G je silne regulárny graf s parametrami (n, k, λ, μ), ktorý má automorfizmus ϕ. Nech H je indukovaný podgraf grafu G, ktorý pozostáva z orbitných vrcholov. Potom platí:*

$$\delta(H) \geq k - max(\lambda, \mu), \text{ kde } \delta(H) \text{ je minimálny stupeň vrchola v grafe } H$$

Dôkaz. Nech x je orbitný vrchol G a nech vrchol y, kde $y = \phi(x)$. Potom nech z je fixný vrchol G, ktorý susedí s vrcholom x. Platí ak $(x, z) \in E(G)$, potom $(\phi(x), \phi(z)) = (y, z) \in E(G)$. To znamená, že existuje vrchol, ktorý susedí s x a zároveň s y. Najviac takýchto vrcholov môže byť $max(\lambda, \mu)$. Graf G je k-regulárny, to znamená, že každý orbitný vrchol musí mať stupeň najmenej $k - max(\lambda, \mu)$. Z toho dostávame:

$$\delta(H) \geq k - max(\lambda, \mu)$$

□

Pomocou nasledujúcej vety vieme zhora ohraničiť počet fixných vrcholov. Zaujímavou vlastnosťou tejto vety je, že nezávisí od veľkosti grupy automorfizmu rádu p. Tieto vety odvodil *Behbahani* vo svojej práci.

Veta 4.6 (Behbahani)**.** *Nech G je silne regulárny graf s parametrami (n, k, λ, μ) a nech $s < r < k$ sú vlastné čísla grafu G. Potom platí:*

$$f \leq n\frac{max(\lambda, \mu)}{k - r}$$

Dôkaz vety neuvádzame a je možné ho nájsť v *Behbahaniho* práci [Beh09]. Dôkaz vety vychádza zo spektrálnej vety lineárnej algebry a idempotentných matíc silne regulárneho grafu.

Veta 4.7 (Behbahani)**.** *Nech G je silne regulárny graf s parametrami (n, k, λ, μ). Potom platí:*

$$f \leq n - \frac{k^2 - k}{max(\lambda, \mu)} + 2k - max(\lambda, \mu) - 2$$

Dôkaz. Nech A je matica susednosti grafu G a nech B je podmatica veľkosti $m \times m$ matice A zodpovedajúca orbitným riadkom a stĺpcom. Z lemy 4.5 vieme, že každý stĺpec má aspoň $k - max(\lambda, \mu)$ jednotiek a zároveň každé dva vrcholy majú najviac $max(\lambda, \mu)$ spoločných susedov. Z toho dostávame vzťah:

$$B^2 \leq (k - max(\lambda, \mu))I + max(\lambda, \mu)J \tag{4.6}$$

Maticu B^2 vieme vypočítať dvoma spôsobmi. Prvý spôsob vychádza z toho, že matica A je symetrická, a preto platí $B^2 = BB^T$. Nech r je ľubovoľný stĺpec v matici B. Podľa lemy 4.5 má aspoň $(k - max(\lambda, \mu))$ jednotiek. Počítaním neusporiadaných dvojíc jednotiek v r stĺpci dostaneme:

$$s = \sum_{1 \leq i,j : i \neq j \leq m} \sum_{r=1}^{m} b_{ir} b_{jr} \geq m(k - max(\lambda, \mu))(k - max(\lambda, \mu) - 1)$$

Použitím rovnice (4.6) dostaneme:

$$s \leq max(\lambda, \mu) m(m-1)$$

Z týchto dvoch vzťahov úpravou dostaneme:

$$m(k - max(\lambda, \mu))(k - max(\lambda, \mu) - 1) \leq max(\lambda, \mu) m(m-1)$$
$$m \geq \frac{k^2 - k}{max(\lambda, \mu)} - 2k + max(\lambda, \mu) + 2$$

Keďže platí $m = n - f$, dostaneme:

$$f \leq n - \frac{k^2 - k}{max(\lambda, \mu)} + 2k - max(\lambda, \mu) - 2$$

□

Pre silne regulárny graf s parametrami (99, 14, 1, 2) z vety 4.6 dostávame $f \leq 18$ a z vety 4.7 $f \leq 32$. *Behbahani* vo svojej práci tvrdil, že pre testy, ktoré robil mu veta 4.6 dávala lepší odhad ako veta 4.7, ale vo všeobecnosti to nevedel dokázať.

Zároveň platia nasledovné vety.

Veta 4.8. *Nech G je silne regulárny graf s parametrami (n, k, λ, μ) s grupou automorfizmom rádu p. Ak $p > k$ a $\mu \neq 0$, potom platí $f = 0$.*

Dôkaz. Z rovnice (4.4) a predpokladu $p > k$ dostávame iba jediné možné riešenie a to $(x_0, x_1, y_0, y_1) = (f - k, k, o, 0)$. Nech u je fixný vrchol G, nech v je orbitný riadok G a nech A je matica susednosti grafu G. Z riešenia sme dostali $y_1 = 0$, z čoho vyplýva, že $A_{uv} = 0$. A podobne ak x je ľubovoľný fixný vrchol G, dostaneme $A_{xv} = 0$ a zároveň ak x je ľubovoľný orbitný vrchol, dostaneme $A_{xu} = 0$. V obidvoch prípadoch vrchol x nie je spoločný sused vrcholov u a v. Zároveň z predpokladu $\mu \neq 0$ vyplýva, že $f = 0$. □

Dôsledok 4.9. *Ak $p > k$ a $\mu \neq 0$, potom n musí byť deliteľný p.*

Veta 4.10. *Nech G je silne regulárny graf s parametrami (n, k, λ, μ) s grupou automorfizmom rádu p. Ak $f < k - 1$ a $p > max(\lambda, \mu)$, potom $f < n/(p + 1)$*

Dôkaz. Z rovnice (4.4) a predpokladov dostávame $py_1 \geq 1$, z čoho vyplýva, $y_1 \geq 1$. Nech matica C' veľkosti $f \times o$ zodpovedá fixným riadkom a orbitným stĺpcom podmatici orbitnej matice. Keďže $y_1 \geq 1$, z toho vyplýva že v každom riadku matice C' je aspoň 1 jednotka. Matica C' má f riadkov, čiže má aspoň f jednotiek. Predpokladajme sporom nech $f > o$. Na základe "holubníkového princípu" *(pigeon hole)* existuje stĺpec v matici C', ktorý má viac jednotiek ako 1. Potom nech sú dva fixné riadky i a j a orbitný stĺpec k také, že $c_{ik}c_{jk} = 1$. Ale zároveň skalárny súčin pre dva fixné riadky v A je menej alebo rovný $max(\lambda, \mu)$, čiže $pc_{ik}c_{jk} \leq max(\lambda, \mu)$. Čo je v spore s predpokladom $p > max(\lambda, \mu)$. Z toho vyplýva, že $f \leq o$ a zároveň $f + po = n$, z toho dostaneme $f < n/(p + 1)$. □

Kapitola 5

Generovanie orbitných matíc s automorfizmom rádu 3

V tejto kapitole navrhneme náš algoritmus pre hľadanie orbitných matíc s grupou automorfizmu rádu 3 metódou úplného prehľadávania. Rôznymi metódami sa potom dá zdvihnúť orbitná matica na maticu susednosti silne regulárneho grafu. *Behbahani* sa vo svojej práci zaoberal orbitnými maticami hlavne s grupou automorfizmu rádu p, kde $p \geq 5$. Dôvodom môže byť, pretože vo svojej práci to nezdôvodnil, že pre $p = 3$ je počet aj veľkosť takýchto matíc veľký a je aj obtiažne ich zdvihnúť. Vo svojej práci sme vychádzali hlavne z popisu algoritmu jeho práce a zároveň navrhli vlastný spôsob testovania na izomorfizmus orbitných matíc, ktorý v práci nespomenul. Cieľom práce je konkrétna implementácia nášho algoritmu vo vhodnom programovacom jazyku. K práci je pribalený zdrojový kód programu napísaný v programovacom jazyku *Scala* [sca15a], v práci popíšeme základné algoritmy formou pseudokódu alebo kódu v *Scale*, ktorý podrobne popíšeme. Vybrali sme si programovací jazyk *Scala* kvôli tomu, že je to moderný jazyk, tzv. "lepšia Java", silne typový jazyk, ktorý umožňuje programovať funkcionálne a na základe toho ľahko distribuovať, t.j. rátať vo viacerých vláknach. Výsledky nášho algoritmu sa nachádzajú v ďalšej kapitole.

5.1 Odhad počtov fixných vrcholov pre orbitné matice s automorfizmom rádu 3

Odhady, ktoré urobil *Behbahani* v časti 4.4 vo svojej práci nevyužívajú automorfizmus a vo všeobecnosti dávajú tesný odhad, ak stupeň k v silne regulárnom grafe je malé číslo. Nás

zaujímajú grafy, kde k je veľké a zároveň $k < n/2$. V tejto časti odvodíme odhady, ktoré dávajú lepšie výsledky pre takéto k.

Predpokladajme ľubovoľný orbitný riadok v orbitnej matici C. Nech d je hodnota na diagonále tohto riadku. Možné hodnoty na diagonále podľa vety 4.2 sú 0 alebo 2. Počet orbitných riadkov vieme ľahko vypočítať pomocou rovnice 4.3 a to $o = (n - f)/3$. Z (4.5) a zároveň zo definície silne regulárneho grafu dostaneme sústavu lineárnych rovníc:

$$\begin{aligned} \bar{y_0} + \bar{y_1} + \bar{y_2} + \bar{y_3} + 1 &= o = \frac{n-f}{3} \\ \bar{y_1} + 2\bar{y_2} + 3\bar{y_3} + x_3 + d &= k \\ \bar{y_2} + 3\bar{y_3} + x_3 &= \mu, \text{ ak } d = 0 \\ \bar{y_2} + 3\bar{y_3} + x_3 + 1 &= \lambda, \text{ ak } d = 2 \end{aligned}$$

Hodnoty $\bar{y_i}$, kde $i = 0, \ldots, 3$ sú počty prvkov i nad orbitnými stĺpcami okrem diagonály a x_3 je počet 3 nad fixnými stĺpcami. Posledné 2 rovnice vyplývajú z definície silne regulárneho grafu, konkrétne z matice susednosti.

Na základe tejto sústavy lineárnych rovníc vieme urobiť 2 odhady pre silne regulárny graf s parametrami (n, k, λ, μ), ak na diagonále je 0 alebo 2.

Ak na diagonále je 0, dostaneme sústavu rovníc:

$$\begin{aligned} \bar{y_0} + \bar{y_1} + \bar{y_2} + \bar{y_3} + 1 &= o = \frac{n-f}{3} \\ \bar{y_1} + 2\bar{y_2} + 3\bar{y_3} + x_3 &= k \\ \bar{y_2} + 3\bar{y_3} + x_3 &= \mu \end{aligned}$$

Úpravou dostaneme:

$$f = n - 3(1 + k - \mu + \bar{y_0} + \bar{y_3})$$

Pre horný odhad fixných vrcholov f môžeme predpokladať, že $\bar{y_0} + \bar{y_3} = 0$. Potom dostaneme odhad:

$$f \leq n - 3(1 + k - \mu) \tag{5.1}$$

Podobne ak na diagonále je 2, dostaneme sústavu rovníc:

$$\begin{aligned} \bar{y_0} + \bar{y_1} + \bar{y_2} + \bar{y_3} + 1 &= o = \frac{n-f}{3} \\ \bar{y_1} + 2\bar{y_2} + 3\bar{y_3} + x_3 &= k \\ \bar{y_2} + 3\bar{y_3} + x_3 + 1 &= \lambda \end{aligned}$$

Úpravou dostaneme:

$$f = n - 3(k - \lambda + \bar{y_0} + \bar{y_3})$$

Pre horný odhad počtu fixných vrcholov f môžeme predpokladať, že $\bar{y_0} + \bar{y_3} = 0$. Potom dostaneme odhad:

$$f \leq n - 3(k - \lambda) \tag{5.2}$$

Obidva odhady vieme zjednotiť do nasledujúcej vety.

Veta 5.1. *Nech G je silne regulárny graf s parametrami (n, k, λ, μ) s automorfizmom rádu 3, kde d je hodnota na diagonále. Potom platí:*

$$f \leq n - 3(1 + k - \mu + \frac{d(d + \mu - \lambda - 3)}{2})$$

Pre konferenčné silne regulárne grafy z časti 2.3 vieme ešte zlepšiť odhad. Vieme, že sú to silne regulárne parametre s jedným parametrom a to $(4\mu + 1, 2\mu, \mu - 1, \mu)$, pričom vlastné čísla sú $k = 2\mu$ a $\frac{1\pm\sqrt{4\mu-1}}{2}$, ktoré nemusia byť celé čísla. Zároveň násobnosti vlastných čísel sú 1, 2μ a 2μ. Dosadením do vety 5.1 dostaneme:

Dôsledok 5.2. *Nech G je konferenčný silne regulárny graf s parametrami (n, k, λ, μ) s automorfizmom rádu 3. Potom platí:*

$$f \leq \mu - 2$$

Tento odhad je oveľa lepší pre konferenčné graf než *Behbahaniho* 4.6, ktorý je väčší ako 2μ. Zároveň ukážeme, že vieme tento odhad pre konferenčné grafy, ktorých vlastné čísla nie sú celé čísla, zlepšiť.

Z lemy v knihe [BH12][Lema 2.3.1] vyplýva, že orbitná matica má rovnaké vlastné čísla ako silne regulárny graf. Silne regulárny graf s parametrami (n, k, λ, μ) má vlastné čísla k, r a s s násobnosťami $1, \bar{f}, \bar{g}$, potom orbitná matica má vlastné čísla k, r a s s násobnosťami $1, f'$ a g', pre ktoré platí $f' \leq \bar{f}$ a zároveň $g' \leq \bar{g}$. Nech orbitná matica je veľkosti $m \times m$, potom platí: $1 + f' + g' = m$. Z vety 1.2 platí:

$$tr(B) = \sum \lambda_i = k + f'r + g's \tag{5.3}$$

Úpravou dostaneme:

$$tr(B) = k + (m - 1)s + f'(r - s) \tag{5.4}$$

Pre konferenčné silne regulárne grafy s parametrami $(4\mu + 1, 2\mu, \mu - 1, \mu)$ platí $\bar{f} = \bar{g}$ a zároveň súčet vlastných čísel $r + s$ je rovný -1. Dosadením do rovnice (5.3) dostaneme:

$$tr(B) = k + f'r + g'(-r - 1) = k - g' + (f' - g')r$$

Vieme, že hodnota $tr(B)$ je celé číslo a zároveň vlastné číslo r nie je celé číslo. Z toho vyplýva, že musí platiť $f' = g'$. Potom m je nepárne číslo, $m = 2f' + 1$. Úpravou dostaneme:

$$tr(B) = k - \frac{m-1}{2}$$

Zároveň veľkosť orbitnej matice je rovný $m = f + o = f + \frac{n-f}{3} = \frac{2f+4\mu+1}{3}$. Úpravou dostaneme:

$$tr(B) = \frac{4\mu - f + 2}{3} = \frac{n-f}{3}$$

Keďže jediné hodnoty na diagonále orbitnej matice môžu byť iba 0 alebo 2, dostávame že $tr(B)$ musí byť párne číslo. Na základe toho vieme sformulovať ďalšiu vetu.

Veta 5.3. *Nech G je konferenčný silne regulárny graf s parametrami (n, k, λ, μ) s automorfizmom rádu 3, ktorého vlastné čísla nie sú celé čísla. Potom platí:*

$$f \equiv n \mod 6$$

Podobne vieme aj pre silne regulárne grafy, ktoré majú vlastné čísla celé čísla, obmedziť počet fixných vrcholov pomocou rovnice 5.4. Úpravou dostaneme:

$$tr(B) \equiv k + (m-1)s \mod (r - s) \tag{5.5}$$

Ukážeme pre konkrétnu sadu parametrov (49, 18, 7, 6), že pre počet fixných vrcholov $f = 13$ neexistujú orbitné matice. Nech $f = 13$, potom $m = 13 + \frac{49-13}{3} = 13 + 12 = 25$. Silne regulárny graf s parametrami (49, 18, 7, 6) má vlastné čísla $r = 4$ a $s = -3$. Naše odhady sú: ak na diagonále 0, $f \leq 10$, ak na diagonále 2, $f \leq 16$ z 5.1. Dosadením parametrov do rovnice (5.5) dostaneme:

$$tr(B) \equiv 18 + 24(-3) \mod (4 - (-3))$$

$$tr(B) \equiv 2 \mod 7$$

Z toho vyplýva, že jediné hodnoty aké môže mať $tr(B)$ sú: $2, 9, 16, 23$ a zároveň musí platiť $tr(B) \leq 24$. Ľahko môžeme vidieť, že na diagonále nemôžu byť samé 2, ale musí byť aj 0. Z nášho odhadu, ak na diagonále je 0 nemôže byť počet fixných vrcholov väčší ako 9. Z toho vyplýva, že neexistuje orbitná matica pre silne regulárny graf s parametrami (49, 18, 7, 6) s $f = 13$. Podobne sa dajú vylúčiť niektoré možnosti na počet pevných bodov aj pre ďalšie sady parametrov.

5.2 Výpočet fixných a orbitných prototypov

Na začiatku algoritmu si pre silne regulárny graf $G = (n, k, \lambda, \mu)$ s danými parametrami a počtom fixných vrcholov f vypočítame prototypy pre fixný riadok a pre orbitný riadok. Pre fixný riadok dostávame sústavu lineárnych rovníc:

$$\begin{aligned} x_0 + x_1 &= f \\ y_0 + y_1 &= o \\ x_1 + 3y_1 &= k \end{aligned} \tag{5.6}$$

Pre orbitný riadok je sústava lineárnych rovníc:

$$\begin{aligned} x_0 + x_3 &= f \\ y_0 + y_1 + y_2 + y_3 &= o \\ x_3 + y_1 + 2y_2 + 3y_3 &= k \\ 3x_3 + y_1 + 4y_2 + 9y_3 &= s_{ii}/3 \end{aligned} \tag{5.7}$$

Z vety 4.2 vyplýva, že na vyrátanie hodnoty s_{ii} môže byť c_{ii} buď 0 alebo 2. Z toho vyplýva, že máme 2 rôzne systémy rovníc pre orbitný riadok podľa hodnoty na diagonále v orbitnej matici. Všetky prototypy pre fixný a orbitný riadok ľahko vieme nájsť vyskúšaním všetkých možných riešení *(brute force)* pomocou spätného vyhľadávania *backtracking*. Uvádzame jednoduchý všeobecný pseudokód 3 na hľadanie všetkých celočíselných (nezáporných) riešení lineárnych rovníc, ktorý ako vstup dostane sústavu lineárnych rovníc, počet premenných a číslo, ktoré reprezentuje maximálne možnú hodnotu premenných v sústave. Výsledok vráti ako množinu všetkých takýchto riešení sústav lineárnych množín. Ak m je počet premenných a max je maximálna hodnota premennej, potom funkcia *Generate* nám vygeneruje všetky možné m-tice, pozostávajúce od 0 po max. Funkcia *SolveEquation* overí každú m-ticu a uloží si riešenie. Vo všeobecnosti tento algoritmus nie je efektívny, ale v našom prípade sú m a max malé čísla.

V prílohe, konkrétne v súbore *GenerateOrbitMatrices.scala*, k tomu prislúchajú funkcie: *solveLinearEquations, generate, solveFixedEq, solveOrbitEq*.

Algoritmus 3 Algoritmus na hľadanie všetkých celočíselných riešení lineárnych rovníc

Vstup: linearSystem, vector, maxInt, numberVariables

Výstup: solutions

```
1:  function GENERATE(solutions: Set[Array[číslo]], maxInt: číslo, depth: číslo, maxDepth:
    číslo)
2:      if depth = maxDepth then
3:          return solutions
4:      else
5:          all ← Set []
6:          for i = 0 → maxInt do
7:              withInt ← Set []
8:              for all sol ← solutions do
9:                  withInt += sol :+ i                ▷ pridá prvok i nakoniec poľa sol
10:             all += withInt
11:         return GENERATE(all, maxInt, depth + 1, maxDepth)
12: function SOLVEEQUATIONS(linearSystem, vector)
13:     gen ← GENERATE(Set [], maxInt, 0, numberVariables)
14:     solutions ← Set []
15:     for all solution ← gen do
16:         if solution je riešenie sústavy lin. rovníc then
17:             solutions += solution
18:     return solutions
```

Na využitie algoritmu 3 na hľadanie všetkých fixných a orbitných prototypov potrebujeme určiť maximálnu hodnotu, akú môžu mať premenné v sústave lineárnych rovníc. Ľahko môžeme pozorovať, že na hľadanie všetkých fixných prototypov máme sústavu lineárnych rovníc (5.6), kde maximálna hodnota môže byť pre x_0, x_1 f a pre y_0, y_1 je o. Podobne na hľadanie všetkých orbitných prototypov máme sústavu lineárnych rovníc (5.7), kde maximálna hodnota pre x_0, x_3 môže byť f a pre y_0, y_1, y_2, y_3 môže byť o. Hodnoty f a o sú malé čísla, a preto náš algoritmus 3 je efektívny. Uvádzame všeobecný pseudokód algoritmu 4, ktorý nájde všetky fixné a orbitné prototypy. Funkcie *SolveFixedEq* a *SolveOrbitEq* sú upravené algoritmy z 3, kde sú pridané predchádzajúce informácie.

Algoritmus 4 Algoritmus na hľadanie všetkých fixných a orbitných prototypov

Vstup: Parametre silne regulárneho grafu (n, k, λ, μ) s f pevnými vrcholmi a o orbitnými vrcholmi

Výstup: fixSolution, orbitSolution0, orbitSolution2

1: **procedure** SOLVELINEAREQUATIONS()
2: linearSystem $\leftarrow [[1,1,0,0],[0,0,1,1],[0,1,0,3]]$
3: vector $\leftarrow [f, o, k]$
4: fixSolution $\leftarrow$ SOLVEFIXEDEQ(linearSystem, vector).filter($x \rightarrow x(0) \neq 0$) ▷ riešenia na konci testujeme $x_0 \geq 1$ pretože na diagonále musí byť 0
5: linearSystem $\leftarrow [[1,1,0,0,0,0],[0,0,1,1,1,1],[0,1,0,1,2,3],[0,3,0,1,4,9]]$
6: sum $\leftarrow 3(k-\mu)+9\mu$
7: vector $\leftarrow [f, o, k, sum/3]$
8: orbitSolution0 $\leftarrow$ SOLVEORBITEQ(linearSystem, vector).filter($x \rightarrow x(2) \neq 0$) ▷ riešenia na konci testujeme $y_0 \geq 1$ pretože na diagonále je 0
9: sum $\leftarrow 3(k-\mu)+9\mu+6(\lambda-\mu)$
10: vector $\leftarrow [f, o, k, sum/3]$
11: orbitSolution2 $\leftarrow$ SOLVEORBITEQ(linearSystem, vector)

Príklad 5.1. *Zoberme si silne regulárny graf s parametrami (15, 6, 1, 3) a nech $f = 3$. Chceme nájsť všetky fixné a orbitné prototypy pre orbitnú maticu rádu 3. Pre fixný prototyp nám z rovnice* (5.6) *vyplýva:*

$$x_0 + x_1 = 3$$

$$y_0 + y_1 = 4$$

$$x_1 + 3y_1 = 6$$

Maximálna hodnota pre x_0 a x_1 je 3 a pre y_0 a y_1 je 4. Pre túto sústavu máme 2 riešenia: $(x_0, x_1, y_0, y_1) = \{(0,3,3,1),(3,0,2,2)\}$. Prvé riešenie lineárnej sústavy rovníc, ale nemôže byť fixný prototyp, pretože musí platiť $x_0 \geq 1$. Preto dostávame iba jeden fixný prototyp pre orbitnú maticu a to $(3,0,2,2)$. Podobne pre orbitný prototyp z rovnice (5.7) vyplýva:

$$x_0 + x_3 = 3$$

$$y_0 + y_1 + y_2 + y_3 = 4$$

$$x_3 + y_1 + 2y_2 + 3y_3 = 6$$

$$3x_3 + y_1 + 4y_2 + 9y_3 = s_{ii}/3$$

A podobne maximálna hodnota pre x_0 a x_3 je 3 a pre y_0, y_1, y_2 a y_3 je 4. Použitím rovnice (4.2) dostaneme: $s_{ii} = 36 - 6c_{ii}$. Pre $c_{ii} = 0$ dostaneme 5 riešení: $(x_0, x_1, y_0, y_1, y_2, y_3) = \{(0,3,1,3,0,0), (1,2,1,2,1,0), (2,1,1,1,2,0), (3,0,1,0,3,0), (3,0,0,3,0,1)\}$. Posledné riešenie, ale nemôže byť orbitný prototyp, pretože musí platiť byť na diagonále 0 a preto $y_0 \geq 1$. Potom dostávame 4 orbitné prototypy pre orbitnú maticu a to $\{(0,3,1,3,0,0), (1,2,1,2,1,0), (2,1,1,1,2,0), (3,0,1,0,3,0)\}$.

5.3 Generovanie orbitnej matice

Ak máme všetky prototypy pre fixné a orbitné riadky, môžeme začať generovať orbitnú maticu. Pre každý riadok orbitnej matice skúšame každý prototyp, pričom najprv začneme generovať fixné riadky a potom orbitné riadky. Úplné prehľadávanie orbitných matíc robíme pomocou spätného prehľadávania *(backtracking)* do hĺbky, to znamená, že generujeme riadky postupne po jednom. Ak daný riadok nespĺňa podmienky, vrátime sa na predchádzajúci riadok a vygenerujeme iný. Celú vygenerovanú orbitnú maticu si uložíme. Všeobecný algoritmus sa podobá algoritmu 1. V prílohe, konkrétne v súbore *GenerateOrbitMatrices.scala*, k tomu prislúchajú funkcie: *generateFixMatrices, generateOrbitMatrix*.

Pre konkrétny prototyp generujeme všetky možné usporiadania riadka. Prvý riadok matice vieme ľahko zostrojiť pomocou prototypu. Zvyšné riadky generujeme rekurzívne spätným prehľadávaním. Po pridaní riadku do matice celú maticu testujeme, či spĺňa podmienky orbitnej matice. Tieto podmienky vieme upraviť tak, aby sme v danom riadku kontrolovali iba hodnoty, ktoré poznáme. Predpokladajme, že chceme vygenerovať hodnoty i. riadku v orbitnej matici. Z toho vyplýva, že poznáme všetky hodnoty riadkov do $i-1$. riadka. Na začiatku berieme do úvahy každý možný prototyp riadku (fixný alebo orbitný), ktorý nám určuje počet hodnôt pre daný riadok. Použitím rovnice 4.1 dostávame vzťah pre $1 \leq j < i$:

$$s_{ji} = \mu n_i n_j + (\lambda - \mu) c_{ji} n_i \tag{5.8}$$

Hodnotu s_{ji} vieme vypočítať pretože hodnoty c_{ji} poznáme. Z vety 4.1 dostávame vzťah:

$$s_{ji} = \sum_{k=1}^{f+o} c_{jk} c_{ik} n_k \tag{5.9}$$

Z týchto vzťahov vieme vytvoriť sústavu lineárnych rovníc, ktoré musí spĺňať náš i. riadok. Čím väčšie je i, tým je sústava lineárnych rovníc väčšia, a preto ho musíme zjednodušiť. Rozhodli

sme sa to urobiť pomocou algoritmu *(Row Echelon form)* [ech15], ktorý upraví sústavu lineárnych rovníc na dolný trojuholníkový tvar, pričom pracujeme len nad racionálnymi číslami. Keďže riešenia tohto systému musia byť celé čísla, ľahko vieme vyhodnotiť kedy sústava môže mať celočíselné riešenie pomocou hrubej sily. Programovací jazyk *Scala* ľahko umožňuje naprogramovať vlastné operátory, a preto sme ľahko implementovali vlastnú triedu *Rational* a všetky operácie vo funkcii *reducedRowEchelonForm* pracujú nad racionálnymi číslami. Algoritmus má časovú zložitosť $\mathcal{O}(n^3)$, čo v našom prípade vyhovuje. Riešenie redukovanej sústavy rovníc hrubou silou *(brute force)* algoritmom 3 je nedostatočné, pretože si vyžaduje veľkú pamäť. To znamená, že nevieme si dopredu vypočítať všetky možnosti pre sústavu lineárnych rovníc a uložiť do premennej. Riešeniu tohto problému je vyrobiť si vlastný *iterátor (Iterator)*. Hlavnou myšlienkou je, že nepočítame všetky možnosti naraz, ale postupne tak, že vieme vyrátať jednu z druhej. Takéto riešenie nie je pamäťovo ani časovo zložité. Uvádzame konkrétnu funkciu v jazyku *Scala* napísanú funkcionálne, ktorý ako vstup dostane pole $pole$ a maximálnu hodnotu $maxInt$ a ako výstup vráti pole, ďalší prvok.

```
def next(pole: List[Int], maxInt: Int): List[Int] = {
    val i = (0 until pole.size).find(i => pole(i) < maxInt - 1)
    i match {
      case Some(i) => List.fill(i)(0) ++
                      pole.updated(i, pole(i) + 1).drop(i)
      case None    => List.fill(pole.size)(0)
    }
  }
```

Vlastný iterátor všetkých možností riešení sústav lineárnych rovníc v Scale

Kolekcia $List[Int]$ je pre nás pole, ktoré obsahuje celé čísla. Prvý riadok najprv vytvorí rozsah *(range)* od 0 po dĺžku poľa, ktorú zároveň prehľadáva a hľadá prvú pozíciu v poli takú, pre ktorú platí daná podmienka a uloží ju do premennej i. Potom sa daný výsledok porovnáva *(match)*, pričom sa využíva pre *Scalu* typický *pattern matching* s *Option* typom [sca15b]. Ak nenašiel žiadnu takú pozíciu v poli, potom vetva prechádza na $None$ (vyhýbame sa $null$) a vrátime vynulované pole, čo znamená že sme preiterovali (prešli) všetky prvky. Ak našiel pozíciu, potom vetva prechádza na $Some(i)$, kde i je pozícia, ktorá spĺňa danú podmienku. Najprv si vytvoríme pomocné vynulované pole veľkosti i. To pole potom zreťazíme s novým poľom, ktoré vznikne zo starého poľa nasledovne. Najprv inkrementujeme o 1 i-ty prvok v poli a potom zahodíme časť poľa od 0 po i. Na začiatku zavoláme funkciu

s vynulovaným poľom s maximálnou hodnotou v cykle opakujeme volanie funkcie $next$ až kým nedostaneme naspäť vynulované pole. Takýmto spôsobom vieme elegantne funkcionálne vygenerovať všetky možné riešenia pre sústavu lineárnych rovníc, pričom ako v ďalšej časti ukážeme vieme to aj ľahko distribuovať.

Pre fixný riadok testujeme, či je podmatica pod fixnými stĺpcami indukovaným grafom Ω silne regulárneho grafu s automorfizmom rádu 3. To znamená, že musí platiť pre každé $u, v \in \Omega$:

$$\begin{aligned} |Fix(x)| &\equiv n \mod 3 \\ d_\Omega(u) &\equiv k \mod 3 \\ |N_\Omega(u) \cap N_\Omega(v)| &\equiv \lambda \mod 3 \text{ ak } u \sim v \\ |N_\Omega(u) \cap N_\Omega(v)| &\equiv \mu \mod 3 \text{ ak } u \not\sim v \end{aligned}$$

Ľahko vieme tieto podmienky kontrolovať pre riešenia sústavy lineárnych rovníc.

Matica susednosti G je symetrická, z toho vyplýva, že pre orbitnú maticu musí platiť pre $i < j$: $c_{ij} = c_{ji}\left(\frac{n_j}{n_i}\right)$. To znamená, že pre každý i. riadok, ktorý vygenerujeme, vieme vypočítať aj i. stĺpec. Z toho vyplýva, že pri generovaní ďalších riadkov môžeme zredukovať počet prototypov, ktoré už nespĺňajú predbežne vyplnený riadok.

5.4 Izomorfizmus

Aby sme negenerovali veľa rovnakých orbitných matíc vzhľadom na izomorfizmus, pretože ich počet rastie exponenciálne, musíme počas generovania zahadzovať matice, ktoré sú izomorfné medzi sebou. Už pre malé silne regulárne grafy, konkrétne $n \geq 29$, hovoríme o miliónoch, čo je náročné na čas aj pamäť na ich vygenerovanie a zároveň by sme ich museli na konci testovať na izomorfizmus, čo by trvalo pri takom množstve veľmi dlho. Preto sme sa inšpirovali algoritmom 2, ktorý predbežne testuje matice na izomorfizmus a zahadzuje tie matice, ktoré sú izomorfné medzi sebou. Dve orbitné matice sú *izomorfné* práve vtedy, keď permutáciou ich riadkov a stĺpcov zároveň dostaneme z jednej matice druhú maticu.

Pre každú orbitnú maticu C veľkosti n definujeme *certifikát* matice ako $\frac{n(n+1)}{2}$-ticu, ktorá vznikne zreťazením hodnôt z hornej trojuholníkovej matice vrátane diagonály riadok po riadku:

$$cert(C) = (c_{11}, c_{12}, \ldots, c_{1n}, c_{22}, c_{23}, \ldots, c_{2n}, \ldots, c_{nn})$$

Definujeme usporiadanie hodnôt nasledovne: $0 < 1 < 2 < 3 <?$, kde $?$ je nedefinovaná hodnota. Podobne definujeme lexikografické usporiadanie aj na n-tice a to nasledovne. Nech $a = (a_1, \ldots, a_n)$ a $b = (b_1, \ldots, b_n)$. Potom $a < b$ práve vtedy, keď existuje index $i \in \{0, \ldots, n-1\}$ taký, že platí:

$$a_1 = b_1, \ldots, a_i = b_i \text{ a zároveň } a_{i+1} < b_{i+1}$$

Potom dve matice $A < B$ práve vtedy, keď $cert(A) < cert(B)$.

Matica A je v *kanonickom tvare* práve vtedy, keď je minimálna vzhľadom na takéto usporiadanie. To znamená:

$$canon(A) = min\{Sym(n)A\} = min\{\{\pi A : \pi \in Sym(n)\}\}$$

Toto je štandardný kanonický tvar používaný v [Bri96], [Mer99] alebo [Deg07].

Príklad 5.2. *Máme 2 matice:*

$$A = \left(\begin{array}{c|ccc} 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 \\ 3 & 1 & 1 & 0 \end{array}\right) \qquad B = \left(\begin{array}{c|ccc} 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 2 \\ 3 & 1 & 0 & 1 \\ 0 & 2 & 1 & 0 \end{array}\right)$$

$$cert(A) = (0, 0, \mathbf{0}, 1, 0, 2, 1, 0, 1, 0)$$

$$cert(B) = (0, 0, \mathbf{1}, 0, 0, 1, 2, 0, 1, 0)$$

Z toho vyplýva, že $A < B$. Dokonca matica A je v kanonickom tvare.

Naším cieľom je počas generovania riadkov orbitnej matice ju mať v kanonickom tvare. Po pridaní riadku do orbitnej matice testujeme maticu na izomorfizmus tak, že permutujeme riadky a stĺpce matice a porovnáme ich certifikáty. Ak po permutovaní vznikne matica s menším certifikátom, môžeme pôvodnú maticu zahodiť. Je zrejmé, že chceme pridávať riadky, ktoré sú lexikografický usporiadané. Uvádzame konkrétny kód v *Scale*, ktorý efektívne permutuje riadky a stĺpce matice zároveň. Ako vstup je pole čísel, premenná $perm$, ktorá obsahuje poradie riadkov (stĺpcov) novej matice a matica mat. Výsledkom funkcie $permuteMatrix$ je matica, ktorej riadky a stĺpce sú permutované podľa premennej $perm$.

```
type Matrix = List[List[Int]]
def permuteMatrixRow(perm: List[Int], mat: Matrix): Matrix ={
  perm.map(i => mat(i))
}

def permuteMatrixCol(perm: List[Int], mat: Matrix): Matrix = {
  val tra = mat.transpose
  perm.map(i => tra(i)).transpose
}

def permuteMatrix(perm: List[Int], mat: Matrix) : Matrix = {
  permuteMatrixRow(perm, (permuteMatrixCol(perm, mat)))
}
```

Permutovanie riadkov a stĺpcov matice v Scale

Kolekcia $List[Int]$ je pre nás pole, ktoré obsahuje celé čísla. Prvý riadok nám definuje maticu ako dvojrozmerné pole celých čísel. Pri zavolaní funkcie $permuteMatrix$ s parametrami $perm$ a mat sa najprv zavolá funkcia $permuteMatrixCol$, t.j. najprv permutuje stĺpce matice, a potom jeho výsledku funkcia $permuteMatrixRow$, t.j. potom permutuje riadky matice. Funkcia $permuteMatrixCol$ najprv maticu transponuje a podľa poľa $perm$ pre každý prvok z poľa priradí riadky transponovanej matice do výslednej matice, ktorú nakoniec transponuje. Podobne aj funkcia $permuteMatrixRow$ podľa poľa $perm$ pre každý prvok z poľa priradí riadky matice do výslednej matice. Takýmto pekným spôsobom vieme jednoducho a efektívne permutovať riadky a stĺpce matice naraz.

Testovanie všetkých možných permutácii je časovo náročné, pretože počet permutácii rastie veľmi rýchlo pre väčšie matice. Riešenie tohto problému je ťažké, pretože chceme predbežne kontrolovať matice na izomorfizmus a zároveň aby tento postup bol efektívny. Preto sme navrhli slabší invariant a to danú maticu testujeme na "slabý" izomorfizmus. Všimneme, že z predchádzajúcich riadkov orbitnej matice vieme rozdeliť dané stĺpce do tried podľa hodnôt a tie len potom permutovať medzi sebou. Tento test je časovo náročný len pre prvé riadky orbitnej matice. Z toho vyplýva, že čím je viac vyplnená orbitná matica, tým je menej permutácii. Z našich skúseností sme zistili, že kontrolovať sa oplatí len prvých 5 riadkov, potom náš test už nie je efektívny. Tento slabší invariant sme prekonzultovali s expertom v danej oblasti *Sven Reichard* [Rei15] počas letnej školy v Tatrách *Summer School in Coherent Configurations, Permutation Groups and Applications in Algebraic Graph Theory 2014* [sch14]. Počas konzultácie nám *Sven Reichard* dal cenné rady, odkazy na literatúru

a v budúcnosti možnú spoluprácu. Počas generovania fixných riadkov v orbitnej matice test rozdeľujeme na dve časti. Najprv permutuje riadky (stĺpce) nad fixnou časťou a potom permutujeme riadky (stĺpce) nad orbitnou časťou. Keď generujeme orbitné riadky stačí nám permutovať riadky (stĺpce) iba nad orbitnou časťou. Samozrejme tento test sa dá vylepšiť (napríklad o krajné prípady) a zároveň negarantuje nám, že dané matice sú v kanonickom tvare. Z našich skúseností pre našu triedu silne regulárnych grafov $n \leq 55$ bol tento test efektívny. Nakoniec výsledné matice musíme otestovať na izomorfizmus, pričom ich počet nie je vysoký a vieme s existujúcimi algoritmami zrátať v rozumnom čase. Na záver tejto časti ukážeme na príklade postupné generovanie orbitnej matice z príkladu 5.1.

Príklad 5.3. *Pre silne regulárny graf s parametrami (15, 6, 1, 3) s* $f = 3$ *máme 1 fixný prototyp* $(3, 0, 2, 2)$ *a 4 orbitné prototypy pre orbitnú maticu a to* $\{(0, 3, 1, 3, 0, 0), (1, 2, 1, 2, 1, 0), (2, 1, 1, 1, 2, 0), (3, 0, 1, 0, 3, 0)\}$. *Prvé 3 riadky v orbitnej matici sú fixné a ostatné 4 sú orbitné. Pre prvý fixný riadok vieme ľahko iba z jedného prototypu vygenerovať riadok a to tak aby bol lexikograficky najmenší, t. j.* $[000|0011]$. *Pre druhý riadok vieme na základe rovnice* (5.8) *a* $c_{12} = 0$ *vyplýva:*

$$s_{12} = 3 - 2c_{12} = 3$$

Zároveň z rovnice (5.9)*:*

$$s_{12} = 3 = \sum_{k=1}^{7} c_{1k}c_{2k}n_k$$

Hodnoty c_{1k} *poznáme a z toho dostávame sústavu lineárnych rovníc:*

$$3c_{26} + 3c_{27} = 3$$

Zároveň vieme, že druhý riadok je fixný riadok, a preto musí byť vygenerovaný z jediného fixného prototypu $(3, 0, 2, 2)$. *Z toho vyplýva, že máme 4 možnosti pre druhý riadok a to:* $[000|0110]$, $[000|1010]$, $[000|0101]$ *a* $[000|1001]$. *Vzhľadom na náš test na "slabý izomorfizmus" rozdelíme predchádzajúci riadok (prvý riadok) na triedy podľa hodnôt. Fixná časť nie je zaujímavá pretože obsahuje samé nuly, skladá sa z jednej triedy, kde sú všetky stĺpce. Orbitná časť má 2 triedy, v 1. triede sú stĺpce* 4 *a* 5 *(obsahujú 0) a v 2. triede* 6 *a* 7 *(obsahujú 1). Teraz nám stačí permutovať prvky v jednotlivých triedach aby sme našli riadok v kanonickom tvare. Namiesto* $4! = 24$ *prvkov dostaneme* $2!\,2! = 16$ *prvkov permutácii. Ľahko vidieť, že iba jeden riadok je v kanonickom tvare a to* $[000|0101]$. *Podobným postupom vieme ukázať, že aj*

3. riadok má iba jeden riadok v kanonickom tvare a to $[000|1001]$. *Ďalší riadok v orbitnej matici je už orbitný, pre ktorý máme 4 prototypy. Z rovnice* (5.8) *dostaneme:*

$$s_{14} = 9 - 6c_{14} = 9$$

$$s_{24} = 9 - 6c_{24} = 9$$

$$s_{34} = 9 - 6c_{34} = 3$$

Zároveň z rovnice (5.9)*:*

$$s_{14} = 3c_{46} + 3c_{47} = 9$$

$$s_{24} = 3c_{45} + 3c_{47} = 9$$

$$s_{34} = 3c_{44} + 3c_{47} = 3$$

Riešenie tejto sústavy lineárnych rovníc je nasledovné: $(c_{44}, c_{45}, c_{46}, c_{47}) = \{(1, 3, 3, 0), (0, 2, 2, 1)\}$. *Na základe prototypov vieme, že 1. riešenie nespĺňa žiaden prototyp, a preto ho môžeme zahodiť. Z toho dostávame, že 4. riadok vyzerá nasledovne* $[003|0221]$. *Podobne pokračujeme až po posledný riadok, kde dostaneme jedinú orbitný maticu:*

$$C = \left(\begin{array}{ccc|cccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 3 & 0 & 2 & 2 & 1 \\ 0 & 3 & 0 & 2 & 0 & 2 & 1 \\ 3 & 0 & 0 & 2 & 2 & 0 & 1 \\ 3 & 3 & 3 & 1 & 1 & 1 & 0 \end{array}\right)$$

5.5 Záverečný test na izomorfizmus

Keďže výsledné matice izomorfné môžu byť, musíme ich na konci algoritmu otestovať. Využívame pri tom už existujúce algoritmy, ktoré efektívne riešia problém izomorfizmu. Najviac známy je program *Nauty* [MP14] napísaný *McKayom* a pre nekomerčné účely zadarmo. Ako vstup sú dané 2 neorientované jednoduché grafy (bez viacnásobných hrán a slučiek) reprezentované zoznamami susedov a ako výstup vracia informáciu o tom, či sú izomorfné alebo nie.

Naše orbitné matice nie sú symetrické a obsahujú hodnoty od 0 po 3. Preto keď chceme aplikovať algoritmy musíme našu maticu upraviť na *multigraf*. *Multigraf* je graf, ktorý obsahuje viacnásobné hrany *(multihrany)* a slučky v grafe. Najprv musíme z orbitnej matice

urobiť symetrickú maticu. Ako ľahko vidieť orbitná matica nie je symetrická len vo fixnej časti, orbitná časť je symetrická. To vieme upraviť tak, že vo fixnej časti nad orbitnými stĺpcami prvky s hodnotou 1 upravíme na nami zvolenou hodnotou, napríklad 5 a vo orbitnej časti nad fixnými stĺpcami prvky s hodnotou 3 upravíme už zvolenou hodnotou 5.

Príklad 5.4. *Nech C je orbitná matica z príkladu 5.3. Z nej zostrojíme maticu C', ktorá bude symetrická.*

$$C = \left(\begin{array}{ccc|cccc} 0&0&0&0&0&1&1\\ 0&0&0&0&1&0&1\\ 0&0&0&1&0&0&1\\ \hline 0&0&3&0&2&2&1\\ 0&3&0&2&0&2&1\\ 3&0&0&2&2&0&1\\ 3&3&3&1&1&1&0 \end{array}\right) \qquad C' = \left(\begin{array}{ccc|cccc} 0&0&0&0&0&5&5\\ 0&0&0&0&5&0&5\\ 0&0&0&5&0&0&5\\ \hline 0&0&5&0&2&2&1\\ 0&5&0&2&0&2&1\\ 5&0&0&2&2&0&1\\ 5&5&5&1&1&1&0 \end{array}\right)$$

Takto upravená orbitná matica prislúcha matice susednosti nejakého multigrafu. Na riešenie problému izomorfizmu multigrafov existujú známe algoritmy, priamo v *Nauty* ale neexistujú. Preto sme skúsili ostatné existujúce programy (knižnice) *Networkx* [net15] a *Graph-tools* [gra15], ktoré majú implementovaný algoritmus *VF2*. Programy *Networkx* a *Graph-tools* sú napísané v programovacom jazyku *Python* [pyt15]. V obidvoch prípadoch nám z neznámych príčin algoritmy zle ukazovali výsledky, t.j. niektoré grafy, ktoré boli neizomorfné ich algoritmus vyhodnotil, že izomorfné sú (pričom ich poradie vrcholov *(mapping)* nesedel) alebo program nečakane spadol. Tieto prípady sa vyskytovali iba v špecifických prípadoch a dlho nám trvalo, kým sme objavili túto chybu. Po konzultácii na fórach sme pravdepodobne objavili nejaké chyby pri implementácii, alebo sme používali inú interpretáciu ako oni. Po zlej skúsenosti neodporúčame používať tieto programy na hľadanie izomorfizmu multigrafov.

Vrátili sme sa naspäť k overenému programu *Nauty*, kde nám poskytol *M. Ziv-Av* [ZA15] funkcie napísané v *GAP* [GAP15], ktorý používa knižnicu *GRAPE* [Soi15]. Hlavnou myšlienkou je ofarbenie orbitnej matice na jednoduchý graf, kde aplikujeme program *Nauty*. V prílohe k tomu prislúcha súbor *ziv-av.g*. Ako vstup sú dané dve orbitné matice, konkrétne riadkové orbitné matice R, ako výsledok vracia informáciu o izomorfizme. V programe *GAP* sa nám ťažšie pracovalo, preto sme prešli na *Sage* [sag15], ktorý má syntax podobnú programovaciemu jazyku *Python* a vieme používať všetky funkcie z *GAP*.

Keď máme zoznam výsledných orbitných matíc a funkciu, ktorá kontroluje 2 orbitné matice, skúšanie všetkých dvojíc nie je efektívne. Preto používame obmenu *Eratostenového sita*. Ak zistíme, že daný graf je izomorfný s nejakým skôr vygenerovaným grafom, zapamätáme si to a testujeme iba dvojice orbitných matíc, ktorých stav ešte nepoznáme. Počet testov je takto zhruba počet orbitných matíc krát počet tried neizomorfných orbitných matíc. Ak počet výsledných orbitných matíc je veľký, rádovo 10 000, môžeme orbitné matice rozdeliť do tried podľa hodnôt ľahko zrátateľných invariantov, napríklad podľa hodnôt na diagonále, a potom iba v týchto triedach kontrolovať izomorfizmus. V prílohe k tomu prislúcha súbory *Test.sage* a *ziv-av.g*, ktorý algoritmus externe volá na konci.

5.6 Distribuovaný výpočet

Ako sme už skôr spomínali, naša implementácia algoritmu a programovací jazyk *Scala* nám ho umožnuje ľahko distribovať pomocou *Aktor model (Actor model)* [Agh86]. *Aktor model* je matematický model konkurentného výpočtu *concurrent computation*, ktorý používa aktorov *actor* ako základnú jednotku. Aktory komunikujú medzi sebou pomocou správ, tie môžu byť synchronizované alebo nesynchronizované. Každý aktor príjma a posiela správy a na základe nich môže meniť svoje správanie (volať lokálne metódy). Zároveň každý aktor vie vytvárať nových aktorov alebo sám zaniknúť. Ľahko si to vieme predstaviť aktorov ako ľudí, ktorí riešia nejaký problém a jedinou možnou komunikáciou je dialóg medzi ľuďmi (posielanie správ). Tento model je dlho známy a využívajú ho rôzne programovacie jazyky, napr. *Erlang* [erl15], ktorý sa používa masívne v distribuovaných databázach a webových serveroch.

Programovací jazyk *Scala* má vlastnú implementáciu Actor modelu, knižnica *Akka* [akk15], ktorú sme aplikovali na náš algoritmus. Náš algoritmus bol navrhnutý tak, aby každé spätné volanie *(backtrack)* bolo sebestačné. To znamená, že nepoužíva žiadne zdielané premenné, vystačí si s premennými, ktoré dostane v argumentoch. Na začiatku náš algoritmus vytvorí jedného hlavného aktora, ktorý zráta prototypy a na základe nich vytvorí nových aktorov, ktorý začnú generovať orbitnú maticu (v argumentoch dostanú informácie o prototypoch). Každý aktor potom skúša konkrétny riadok, ak sa mu to podarí, vytvorí nových aktorov so svojim riešením na ďalší riadok a sám zanikne. Ak sa mu to nepodarí, zanikne. Ak aktorovi sa podarí nájsť výslednú maticu, pošle správu (orbitnú maticu) hlavnému aktorovi, ktorý si výsledok uloží a zanikne. Program sa zastaví práve vtedy, keď zostane iba

hlavný aktor aktívny. Keďže v jednom momente môžeme mať aktívnych aj milión aktorov, v akom poradí a na ktorom procesore bežia nám zabezpečuje knižnica, ktorá to robí optimálne. To znamená, že vyťažuje naplno všetky procesory (prípadne podľa konfigurácie iba nejaký počet procesorov) a pamäť efektívne rieši "zber odpadkov" *(garbage collector)*. Podobným spôsobom sa dá pomocou konfigurácie napojiť na tento model viacero počítačov a tak zvýšiť výpočtový výkon. Implementácia aktor modelu je oveľa jednoduchšia než tradičný viac vláknový prístup, kde by sme museli riešiť problém synchronizácie. Ďalšou výhodou bolo, že sme nemuseli náš pôvodný algoritmus veľmi upravovať. V prílohe prislúcha k tejto časti súbor *GenerateOrbitMatrixAkka.scala*.

Kapitola 6

Výsledky práce

Popísaný algoritmus v predchádzajúcej kapitole sme aplikovali na sady parametrov silne regulárnych grafov do 55 vrcholov. Tieto výsledky spracoval školiteľ *M. Mačaj*, ktorý orbitné matice zdvihol *(lift)* na matice susednosti silne regulárnych grafov vlastnou metódou. Program bol spúšťaný na 4 rôznych zariadeniach: notebook s konfiguračnou zostavou Intel Core i3 350M, RAM 4GB, HDD 500GB 7200 otáčok, počítač s konfiguračnou zostavou Intel Core i3-2130, RAM 8GB, SSD 32 GB, počítač s konfiguračnou zostavou Intel Core i5-3570k, RAM 8GB, HDD 2TB 7200 otáčok a počítač s konfiguračnou zostavou Intel Core i7-4790, RAM 8GB, SSD 128 GB.

V prvej časti ukážeme vypočítané počty fixných vrcholov pre dané sady parametrov, pričom sme pri tom využili vety z časti 4.4 a 5.1. V ďalšej časti sme aplikovali náš algoritmus na sady parametrov s úplnou klasifikáciou. Zoznamy grafov sme získali od *E. Spence*, *M. Behbahani*, *C. Lam* a *P. R. J. Östergård* [Spe15b] [BLO12]. Našli sme všetky grafy, ktoré sme hľadali. V ďalšej časti sme aplikovali náš algoritmus na sady parametrov bez úplnej klasifikácie. Z doteraz známych grafov sme nielen našli všetky grafy, ale sme aj skompletizovali grafy s automorfizmom rádu 3. Na záver zhrnieme súčasný stav a možné plány do budúcna.

Konkrétne dáta a výsledky je možné si pozrieť na našej stránke [KM15] alebo je ich možné nájsť v prílohe.

6.1 Výpočet počtu fixných vrcholov

Zo stránky [Srg15] sme získali zoznam sád parametrov silne regulárnych grafov, ktoré sú buď kompletne klasifikované, bez úplnej klasifikácie alebo ich existencia nie je známa. Na

základe viet z časti 4.4 a 5.1 sme vypočítali počty fixných vrcholov, pre ktoré sa oplatí spustiť náš program. Výsledky sme rozdelili do dvoch tabuliek. V prvej tabuľke sú výsledky pre konferenčné grafy, kde sme využili lepší odhad vďaka vete 5.2. Na sady parametrov okrem (49, 24, 11, 12) a (25, 12, 5, 6) môžeme aplikovať vetu 5.3, pretože sú to neintegrálne grafy, t. j. vlastné čísla nie sú celé čísla. V druhej sú ostatné (integrálne) grafy, kde sa dá použiť iba veta 5.1 a zároveň vieme pre niektoré počty ukázať, že neexistujú. Ako môžeme pozorovať v obidvoch prípadoch boli *Behbahaniho* odhady oveľa horšie okrem sady parametrov (45, 12, 3, 3). *Vysvetlivky k tabuľke:* n, k, λ a μ sú parametre silne regulárneho grafu, ! kompletná klasifikácia, $+$ neúplná klasifikácia, $?$ existencia nie je známa.

n	k	λ	μ	počet grafov	Behbahaniho odhad	náš odhad
25	12	5	6	15!	≤ 16	1, 4
29	14	6	7	41!	≤ 17	5
37	18	8	9	+	≤ 21	1, 7
41	20	9	10	+	≤ 23	5
45	22	10	11	+	≤ 25	3, 9
49	24	11	12	+	≤ 28	1, 4, 7, 10
53	26	12	13	+	≤ 30	5, 11
61	30	14	15	+	≤ 34	1, 7, 13
65	32	15	16	?	≤ 36	5, 11

Tabuľka 6.1: Počet fixných vrcholov pre konferenčné grafy

n	k	λ	μ	počet grafov	Behbahaniho odhad	náš odhad
26	10	3	4	10!	≤ 13	2, 5
28	12	6	4	4!	≤ 21	1, 4, 7, 10
35	16	6	8	3854!	≤ 20	2, 5, 8
36	14	4	6	180!	≤ 18	0, 3, 6, 9
36	15	6	6	32548!	≤ 18	0, 3, 6, 9
40	12	2	4	28!	≤ 16	1, 4, 7, 10, 13
45	12	3	3	78!	≤ 15	0, 3, 6, 9, 18
49	18	7	6	+	≤ 24	1, 4, 7, 10
50	21	8	9	+	≤ 25	2, 5, 8, 11
57	24	11	9	+	≤ 33	0, 3, 6, 9, 15
63	30	13	15	+	≤ 35	0, 3, 6, 9, 12, 15

Tabuľka 6.2: Počet fixných vrcholov pre integrálne grafy

6.2 Výsledky programu pre sady parametrov SRG so známou úplnou klasifikáciou

Aby sme overili náš algoritmus, spustili sme náš program na sadu parametrov silne regulárnych grafov, pre ktoré je známa úplná klasifikácia. V prvom rade, aby sme mohli naše výsledky skontrolovať s už známymi, sme si zo stránky *E. Spence* [Spe15b] stiahli zoznamy silne regulárnych grafov. Z nich sme vybrali tie grafy, ktoré majú grupu automorfizmu rádu 3 pomocou známych algoritmov v *Sage*. V prílohe je možné nájsť program *SRGAut3.sage*. Pomocou nich sme si aj vyrátali počet fixných vrcholov a skontrolovali s nami vypočítanými počtami, program *FixPoints.sage*. Výsledky sme zahrnuli do nasledujúcej tabuľky:

n	k	λ	μ	počet všetkých grafov	počet grafov s automorfizmom rádu 3
25	12	5	6	15	9
26	10	3	4	10	6
28	12	6	4	4	4
29	14	6	7	41	12
35	16	6	8	3 854	140
36	14	4	6	180	41
36	15	6	6	32 548	344
40	12	2	4	28	20
45	12	3	3	78	44

Tabuľka 6.3: Sady parametrov SRG s úplnou klasifikáciou

Aplikovali sme náš algoritmus na tieto sady parametrov silne regulárneho grafu. Výpočet pre jednotlivé sady parametrov netrval viac ako hodinu. Pozorovali sme, že výpočet pre väčšie počty fixných vrcholov trval dlhšie ako pre menšie. Dôvodom môže byť, že náš "slabý" test na izomorfizmus trvá dlhšie, pretože delenie do tried nie je príliš efektívne. Zároveň sme si všimli, že počty orbitných matíc pre väčší počet fixných vrcholov klesá. Vyskúšali sme hľadať orbitné matice pre komplementárne grafy pre kontrolu aj pre porovnanie doby výpočtu. Pozorovali sme, že doba výpočtu je približne rovnaká. V prílohe aj na stránke uvádzame najzaujímavejšiu sadu parametrov a to (35, 16, 6, 8), kde komplementárny graf je s parametrami (35, 18, 9, 9). Výsledky sme zaznamenali do tabuľky 6.4, kde ako sme spomínali, školiteľ zdvihol orbitné matice na matice susednosti vlastnou metódou.

n	k	λ	μ	Počet orbitných matíc (počet fixných vrcholov)	Počet nájdených grafov z orbitných matíc
25	12	5	6	9 (1) 4 (4)	9
26	10	3	4	3 (2), 1 (5)	6
28	12	6	4	4 (1), 2 (10)	4
29	14	6	7	1 (5)	12
35	16	6	8	18 (2), 3 (5), 1 (8)	140
35	18	9	9	18 (2), 3 (5), 1 (8)	140
36	14	4	6	10 (0), 3 (3), 1 (9)	41
36	15	6	6	13 (0), 30 (3), 4 (6), 3 (9)	344
40	12	2	4	9 (1), 5 (4), 1 (7), 1 (13)	20
45	12	3	3	5 (3), 6 (6), 2 (9)	44

Tabuľka 6.4: Orbitné matice a z nich grafy pre sady parametrov SRG s úplnou klasifikáciou

Ako vidíme, našli sme všetky grafy, ktoré sme hľadali. Keďže výpočet netrval dlho, vytvorili sme si *Unit Testy*, ktoré pri každej veľkej zmene algoritmu automaticky testujú funkcionalitu algoritmu na týchto sadách parametrov. V prílohe sa nachádzajú v priečinku *Test*.

6.3 Výsledky programu na sady parametrov SRG bez úplnej klasifikácie

Na základe úspešného otestovania nášho programu na sadách parametrov silne regulárnych grafov s úplnou klasifikáciou sme spustili náš algoritmus na sadách parametrov bez úplnej klasifikácie. Našou prioritou boli sady parametrov do 55 vrcholov, a keďže algoritmu fungoval úspešne, vyskúšali sme ho aplikovať na ďalšie sady parametrov. Pre kontrolu sme získali zoznamy doteraz známych grafov (nie sú úplné) od *E. Spence*, *M. Behbahani*, *C. Lam* a *P. R. J. Östergård* [Spe15b] [BLO12]. Podobne sme z nich vybrali tie grafy, ktoré majú grupu automorfizmu rádu 3 pomocou známych algoritmov. Výsledky sme zahrnuli do nasledujúcej tabuľky:

n	k	λ	μ	počet doteraz známych grafov	počet známych $\mathbb{Z}_3$-grafov
37	18	8	9	6 760+	31+
41	20	9	10	120+	0+
45	22	10	11	2 014+	172+
49	18	7	6	785+	63+
49	24	11	12	342+	169+
50	14	4	6	1 543+	318+
53	26	12	13	+	+
65	32	15	16	?	?

Tabuľka 6.5: Sady parametrov SRG bez úplnej klasifikácie

Aplikovali sme náš algoritmus na tieto sady parametrov silne regulárnych grafov. Výpočet trval pre jednotlivé sady parametrov rôzne doby. Pre niektoré sady parametrov výpočet trval menej než deň pre iné (hlavne konferenčné) výpočet trval aj mesiac. Výsledky sme zhrnuli do tabuľky 6.6. Ako vidíme, počet orbitných matíc narastá. Ak neexistujú žiadne orbitné matice pre konkrétnu sadu parametrov a počet fixných vrcholov, potom sme ich v tabuľke vynechali.

n	k	λ	μ	Počet orbitných matíc (počet fixných vrcholov)	Počet všetkých $\mathbb{Z}_3$-grafov	Nový počet známych grafov
37	18	8	9	18(1)	37	6802+
41	20	9	10	18(5)	264	5214+
45	22	10	11	7(9)	11 536	51 104+
49	18	7	6	595(1), 107(4), 4(7)	243	1 471+
49	24	11	12	5 029(1), 124(4), 40(7)	4 295	702 395+
50	14	4	6	5 043(2), 106(5), 35(8)	4 353	87 408+
53	26	12	13	1 229(5)	4 438	15 122+
65	32	15	16	302*(5)	?	?

Tabuľka 6.6: Orbitné matice a z nich grafy pre sady parametrov SRG bez úplnej klasifikácie

Školiteľovi sa podarilo tieto orbitné matice zdvihnúť na maticu susednosti a ako môžeme vidieť v tabuľke 6.6 našli sme všetky doteraz známe grafy a navyše sme kompletne klasifikovali všetke silne regulárne grafy s automorfizmom rádu 3 do 55 vrcholov. Zároveň sa školiteľovi sa

podarilo získať ďalšie grafy pomocou rôznych metód *switching* zo získaných grafov [BLO12] [BH12]. Ako vidíme pre sady parametrov (49, 24, 11, 12) a (50, 14, 4, 6) sú počty orbitných matíc podobné. Nie je to náhoda, súvisí to s *regulárnymi 2-grafmi* [Spe15a]. Školiteľovi sa na základe týchto výsledkov podarilo rozšíriť doteraz známe zoznamy regulárnych 2-grafov.

Skúšali sme spustiť náš program na sadu parametrov (65, 32, 15, 16), kde nie je známa ani možná existencia grafu. Program ani po mesiaci nedokončil svoj výpočet, predbežne sme našli 302 orbitných matíc. Školiteľovi sa nepodarilo ich zdvihnúť na maticu susednosti, odhadol že výpočet je príliš náročný.

6.4 Súčasný stav a plány do budúcna

Momentálne sme spustili náš program na ďalšie sady parametrov silne regulárnych grafov a to (57, 24, 11, 9) a (63, 30, 13, 15). Pre sadu parametrov (57, 24, 11, 9) s počtom fixných vrcholov 6 a 9 sa nám už podarilo nájsť grafy a našli sme zatiaľ všetky, ktoré našiel *M. Behbahani*, *C. Lam* a *P. R. J. Östergård* [BLO12].

Ako sme zistili pre väčšie sady parametrov silne regulárnych grafov, počet orbitných matíc prudko rastie. Pri súčasnom stave a implementácii algoritmu nevieme dopredu určiť, ako dlho bude daný výpočet trvať. V budúcnosti by sme to chceli vylepšiť pomocou odhadov a zároveň zlepšiť náš "slabý" test na izomorfizmus. Ďalším cieľom by bolo navrhnúť lepší (univerzálny) distribuovaný model na výpočet, kde by sme chceli prepojiť počítače do spoločnej siete pomocou knižnice *Akka*. Takto navrhnutý model by nebol náchylný na výpadok prúdu, ktorý nám spôsobil menšie problémy a zároveň by sme vedeli ľahko pripojiť ďalšie počítače, prípadne pozastaviť výpočet na počítačoch v sieti.

Záver

V práci sme sa venovali konštruktívnej enumerácii grafov, konkrétne silne regulárnym grafom. Hlavným výsledkom práce je vytvorenie programu na hľadanie potencionálnych orbitných matíc silne regulárnych grafov s automorfizmom rádu 3. Pomocou tohto programu sme vytvorili úplný zoznam potencionálnych orbitných matíc silne regulárnych grafov s parametrami (37, 18, 8, 9), (41, 20, 9, 10), (45, 22, 10, 11), (49, 18, 7, 6), (49, 24, 11, 12), (50, 21, 8, 9) a (53, 26, 12, 13), pre ktoré nie je známa úplna klasifikácia. Z týchto orbitných matíc boli následne vytvorené úplné zoznamy silne regulárnych grafov do 55 vrcholov s automorfizmom rádu 3, ako aj podstatne rozšírené zoznamy známych silne regulárnych grafov do 55 vrcholov, resp. regulárnych 2-grafov do 60 vrcholov. V budúcnosti by sme sa chceli pozrieť na ďalšie sady parametrov bez úplnej klasifikácie.

Literatúra

[Agh86] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA, USA, 1986.

[akk15] Akka. `http://akka.io/`, 2015.

[Bab14] László Babai. On the automorphism groups of strongly regular graphs. `http://people.cs.uchicago.edu/~laci/papers/14itcs.pdf`, 2014.

[Beh09] Majid Behbahani. *On strongly regular graphs*. PhD thesis, 2009.

[BH12] Andries E. Brouwer and Willem H. Haemers. *Spectra of graphs*. Springer, 2012.

[BLO12] Majid Behbahani, Clement Lam, and Patric R.J. Ostergård. On triple systems and strongly regular graphs. *Journal of Combinatorial Theory, Series A*, 119(7):1414 – 1426, 2012.

[Bri96] Gunnar Brinkmann. Fast generation of cubic graphs. *Journal of Graph Theory*, 23(2):139–149, 1996.

[Dec03] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[Deg07] Jan Degraer. *Isomorph-free exhaustive generation algorithms for association schemes*. PhD thesis, 2007.

[ech15] Reduced row echelon form. `http://rosettacode.org/wiki/Reduced_row_echelon_form`, 2015.

[erl15] Erlang programming language. `http://www.erlang.org/`, 2015.

[GAP15] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.7.7*, 2015.

[GKL96] R. Grund, A. Kerber, and R. Laue. Construction of discrete structures, especially isomers. *Discrete Applied Mathematics*, 67(1–3):115 – 126, 1996. Chemistry and Discrete Mathematics.

[gra15] Graph-tool. `https://graph-tool.skewed.de/`, 2015.

[HE79] Robert M. Haralick and Gordon L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'79, pages 356–364, San Francisco, CA, USA, 1979. Morgan Kaufmann Publishers Inc.

[JJ14] Thomas G. Wong Jonatan Janmark, David A. Meyer. Global symmetry is unnecessary for fast quantum search. `http://arxiv.org/abs/1403.2228`, 2014.

[KM15] Anton Kovaľ and Martin Mačaj. Silne regulárne grafy s grupou automorfizmu rádu 3, 2015. `http://davinci.fmph.uniba.sk/~koval13/diplomovka/srg.html`.

[KO04] Petteri Kaski and Patric R. Ostergård. The steiner triple systems of order 19. *MATH. COMP*, 73, 2004.

[KO05] Petteri Kaski and Patric R. J. Östergård. *Classification Algorithms for Codes and Designs (Algorithms and Computation in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[Kov13] A. Kovaľ. Spektrálna teória grafov Bakalárska práca. `http://oldwww.dcs.fmph.uniba.sk/bakalarky/registracia/getfile.php/bp.pdf?id=231&fid=455&type=application%2Fpdf`, 2013.

[Kum92] Vipin Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Mag.*, 13(1):32–44, April 1992.

[Mer99] Markus Meringer. Fast generation of regular graphs and construction of cages. *Journal of Graph Theory*, 30(2):137–146, 1999.

[MMM07] Brendan D. McKay, Alison Meynert, and Wendy Myrvold. Small latin squares, quasigroups, and loops. *Journal of Combinatorial Designs*, 15(2):98–119, 2007.

[MP14] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, 60(0):94 – 112, 2014.

[MW91] Brendan D. McKay and Nicholas C. Wormald. Asymptotic enumeration by degree sequence of graphs with degrees o($n^{1/2}$). *Combinatorica*, 11:369–382, 1991.

[net15] Networkx. `https://networkx.github.io/`, 2015.

[Pal33] R.E.A.C. Paley. On orthogonal matrices. *J. Math. Phys.*, 12:311–320, 1933.

[pyt15] Python. `https://www.python.org/`, 2015.

[Rei15] Sven Reichard. `https://tu-dresden.de/Members/sven.reichard`, 2015.

[sag15] Sagemath - opensource mathematical sowftware system. `http://www.sagemath.org/`, 2015.

[sca15a] The scala programming language. `http://scala-lang.org/`, 2015.

[sca15b] Scala standard library 2.11.6 - option. `http://www.scala-lang.org/api/current/index.html#scala.Option`, 2015.

[Sch93] Bernd Schmalz. The t-designs with prescribed automorphism group, new simple 6-designs. *Journal of Combinatorial Designs*, 1(2):125–170, 1993.

[sch14] Summer school in coherent configurations, permutation groups and applications in algebraic graph theory 2014. `http://esf.umb.sk:8080/sschool14/`, 2014.

[Ser03] Akos Seress. *Permutation Group Algorithms*. Cambridge University Press, 2003.

[Soi15] Leonard H. Soicher. Grape (version 4.6.1) package for gap, 2015. `http://www.maths.qmul.ac.uk/~leonard/grape/`.

[Spe15a] Edward Spence. Conference two-graphs, 2015. `http://www.maths.gla.ac.uk/~es/twograph/conf2Graph.php`.

[Spe15b] Edward Spence. Strongly regular graphs on at most 64 vertices, 2015. `http://www.maths.gla.ac.uk/~es/srgraphs.php`.

[Srg15] Parameters of strongly regular graphs. http://www.win.tue.nl/~aeb/graphs/srg/srgtab.html, 2015.

[Ste14] Dragan Stevanovic. Energy of graphs. http://www.public.iastate.edu/~lhogben/energyS, 2014.

[Tom11] Eduard Toman. Enumerácia diskrétnych štruktúr. http://new.dcs.fmph.uniba.sk/files/texty/eds.pdf, 2011.

[ZA15] Matan Ziv-Av. http://www.cs.bgu.ac.il/~zivav/, 2015.

[Zla11] Pavol Zlatoš. Lineárna algebra a geometra. http://thales.doa.fmph.uniba.sk/zlatos/la/LAG_A4.pdf, 2011.