Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Jan Kenda

Hitro 3-barvanje omejenih ravninskih grafov

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentor: prof. dr. Gašper Fijavž

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Metoda prenosa naboja je postopek za barvanje grafovskih struktur v ravninskih grafih. Salavatipour (The Discharging Method in Practice, 2006) predstavlja dokaz, da lahko ravninske grafe brez ciklov dolžin 4–9 obarvamo s tremi barvami. Njegov dokaz je moč prepisati v kvadratičen algoritem za 3-barvanje. V delu predstavite njegov postopek, izolirajte časovno kritičen del in izdelajte linearen algoritem za 3-barvanje takšnih grafov.

Zahvaljujem se svojemu mentorju, prof. dr. Gašperju Fijavžu, za potrpežljivost, za njegov čas, odlične nasvete in posvete.

Zahvaljujem se tudi staršem za potrpežljivost in ker so mi vedno stali ob strani.

Kazalo

Povzetek

Naslov: Hitro 3-barvanje omejenih ravninskih grafov

Avtor: Jan Kenda

V delu obravnavamo problem 3-barvanja ravninskih grafov brez ciklov dolžin med 4 in 9. Salavatipour (The Discharging Method in Practice, 2006) je skupaj z dokazom 3-obarvljivosti teh grafov implicitno zapisal tudi kvadratičen algoritem 3-barvanja. V delu predstavimo postopek prenosa naboja, ki je osnovna ideja takega algoritma. Hkrati z natančnejšo strukturno analizo pokažemo, da je moč omenjeni algoritem poenostaviti in hkrati pohitriti. Tako izboljšani algoritem je celo linearne časovne zahtevnosti, kar tudi empirično preverimo.

Ključne besede: ravninski grafi, barvanje grafov, metoda prenosa naboja.

Abstract

Title: Efficient 3-coloring of a subclass of planar graphs

Author: Jan Kenda

In the thesis we present the 3-coloring problem of planar graphs without cycles of lengths between 4 and 9. Salavatipour (The Discharging Method in Practice, 2006) has proven that such graphs are 3-colorable and his proof can be directly rewritten as a quadratic-time algorithm. We present the discharging method which is the cornerstone for this algorithm. What is more, we focus on the time-critical part and with a careful analysis show that it is indeed not needed. Thus we invent a linear time 3-coloring algorithm of such graphs and we also evaluate its implementation.

Keywords: planar graphs, graph coloring, discharging method.

Poglavje 1

Uvod

Problem barvanja grafov se je začel z barvanjem zemljevidov. Sega nazaj več kot 150 let, ko je leta 1852 matematik De Morgan poslal pismo svojemu prijatelju, matematiku Hamiltonu. Napisal je, da je njegov študent Guthrie ugotovil, da je možno zemljevid grofij Anglije pobarvati s samo štirimi barvami tako, da nobeni dve sosednji grofiji nista iste barve. S tem se je začel znameniti problem štirih barv, ki so ga matematiki (neuspešno) dokazovali in drug drugemu izpodbijali dokaze več kot 100 let, dokler ni leta 1976 Appelu in Hakenu [1] s pomočjo računalnikov uspelo dokazati, da je mogoče vsak ravninski graf pobarvati z zgolj štirimi barvami.

Vsebina diplomske naloge je barvanje ravninskih grafov brez ciklov dolžin med 4 in 9. Da je take grafe mogoče pobarvati le s tremi barvami, je že pred več kot dvajsetimi leti ugotovil in dokazal Borodin [2], leta kasneje pa je še dokazal, da je mogoče tudi ravninske grafe brez ciklov dolžin 5 in 7, ter brez sosednjih trikotnikov, pobarvati le s tremi barvami [3].

Borodinovi izreki so matematični dokazi obstoja takšnega barvanja, prepisani kot algoritem pa so časovno potratni. Namen naše naloge je poizkusiti, ali je take grafe mogoče učinkovito pobarvati. Za dokaz določenih struktur (podgrafov) smo uporabili metodo prenosa naboja (opisano v poglavju 3). Ta metoda nam ne potrdi zgolj obstoja želenih struktur, marveč nam pomaga tudi pri njihovem iskanju.

Razvili smo algoritem, ki takšne grafe pobarva v linearnem času. Ob tem smo zgradili tudi generator takšnih grafov in tako testirali delovanje algoritma.

Poglavje 2

O grafih in barvanju

2.1 Grafi

Definicija 2.1 *Graf G je urejen par $G = (V, E)$, kjer je V neprazna množica vozlišč, E pa množica povezav med njimi. Vsaka povezava $e \in E$ je par vozlišč $v_1, v_2 \in V$, ki sta začetno in končno vozlišče povezave.*

V usmerjenih grafih je vrstni red vozlišč v povezavi pomemben, v neusmerjenih pa ne. Izpostavimo dve posebni vrsti povezav: v primerih, ko se povezava začne in konča v istem vozlišču, to povezavo imenujemo zanka. Dve povezavi, ki povezujeta isti vozlišči, pa se imenujeta vzporedni povezavi. Neusmerjen graf s končnim številom vozlišč brez zank in vzporednih povezav imenujemo enostavni graf. Problem barvanja vozlišč grafa zahteva, da sta sosednji vozlišči obarvani z različnima barvama. Zato se lahko pri problemih barvanja — in tudi v tem diplomskem delu — omejimo na enostavne grafe.

Definicija 2.2 *Stopnja vozlišča v, pisano $d(v)$, je število sosedov vozlišča v. Dve vozlišči sta sosedi, če med njima obstaja povezava.*

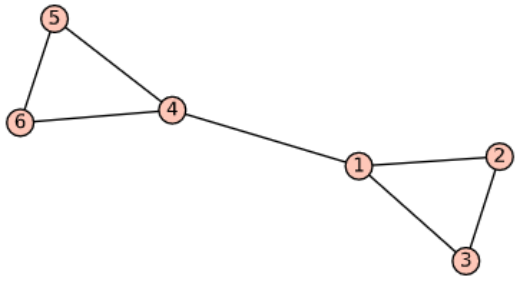
V enostavnem grafu ima vozlišče v stopnjo $d(v)$ in tudi toliko različnih sosedov. *Pot* v grafu je zaporedje različnih vozlišč, v katerem sta vsaki zaporedni vozlišči sosedi. Pot, ki vsebuje samo vozlišča stopenj 1 ali 2, imenujemo *veja*. Pot, ki ima začetno in končno vozlišče isto, pa imenujemo *cikel*. Graf G je

povezan, če za vsak par vozlišč $v_1, v_2 \in G(V)$ obstaja pot z začetkom v v_1 in koncem v v_2. Če graf G ni povezan, njegove maksimalne povezane podgrafe imenujemo *komponente* grafa G.

2.2 Ravninski grafi

Graf je *ravninski*, če ga lahko narišemo tako, da se njegove povezave sekajo le v krajiščih. Takšnemu risanju na ravnino pravimo *vložitev* grafa. Graf, ki je narisan s križišči in je izomorfen ravninskemu grafu, je vseeno ravninski graf, le da njegova vložitev še ni določena. Vsak ravninski graf je mogoče narisati tudi na sfero.

Pri ravninskih grafih se pojavi nov pojem: *lice*. Definicijo G razširimo na $G = (V, E, F)$, kjer je F množica lic. Ko graf vložimo v ravnino, povezave grafa razdelijo to ravnino na več območij. Tem območjem pravimo lica. *Dolžina lica* f, pisano $\ell(f)$, je dolžina obhoda (število povezav), ki lice obdaja. Povezave in vozlišča se lahko na istem licu pojavijo večkrat.



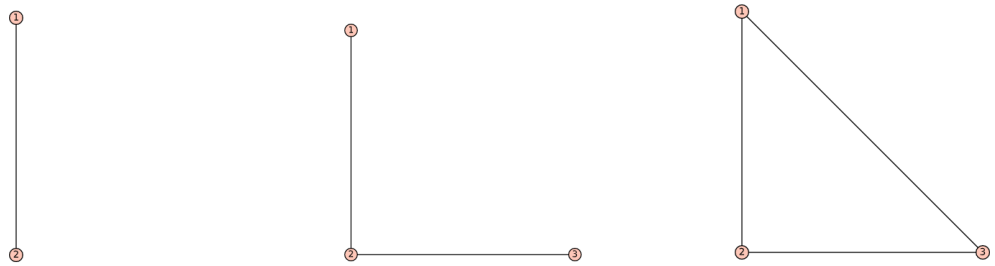
Slika 2.1: Lice dolžine 8, kjer se vozlišči 1 in 4 pojavita dvakrat.

Za vsako vložitev grafa G so njegova lica enolično določena.

Izrek 2.2.1 *Eulerjeva formula za ravninske grafe pravi, da za vsak končen, povezan, vložen ravninski graf G, ki vsebuje v vozlišč, e povezav in f lic, velja*

$$v - e + f = 2. \tag{2.1}$$

Dokaz: začnemo z vpetim drevesom na n vozlišlih, $n-1$ povezavah in z enim samim licem, torej $n-(n-1)+1=2$. Če dodamo novo povezavo med dvema vozliščema, se število povezav poveča za 1, hkrati pa tudi razdeli eno lice na dva dela. Seštevek $v-e+f$ v tem primeru ostane nespremenjen. Po indukciji velja formula za vse povezane ravninske grafe.



Slika 2.2: Dodajanje vozlišča in povezave.

2.3 Barvanje

Definicija 2.3 *Naj bo G graf in k naravno število. Preslikavi $f : V(G) \to \{1, 2, ..., k\}$ pravimo k-barvanje grafa G, če za poljuben par sosednjih vozlišč $v_1, v_2 \in G$ velja $f(v_1) \neq f(v_2)$.*

Definicija pomeni, da če vsakemu vozlišču v grafu priredimo eno izmed k barv, bo to vozlišče drugače obarvano, kot njegovi sosedi. Grafu, ki ga je mogoče pobarvati s k barvami, pravimo k-obarvljiv graf. Seveda je vsak graf na n vozliščih n-obarvljiv — vsako vozlišče pobarvamo z drugačno barvo in problem je rešen. Tako barvanje je neuporabno, zato vpeljemo pojem *kromatičnega števila*. Kromatično število $\chi(G)$ je najmanjše naravno število k, za katero je graf G k-obarvljiv.

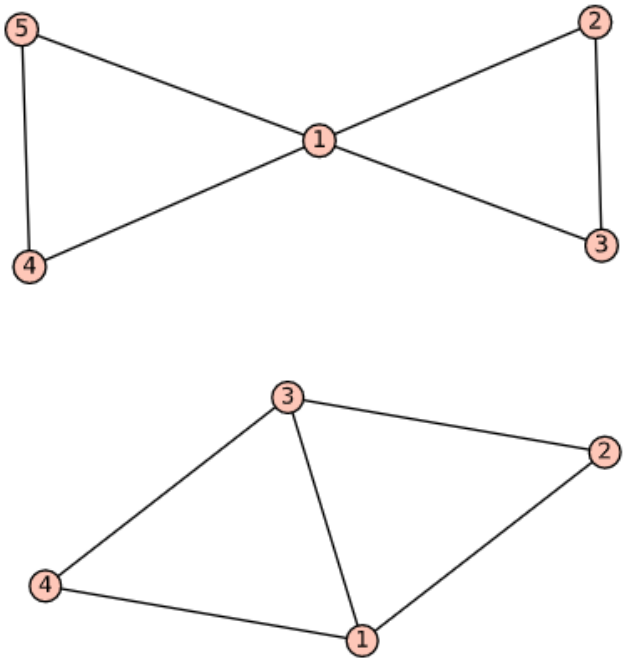
Barvanje nepovezanega grafa je enakovredno barvanju vsake komponente grafa neodvisno od ostalih, saj barve vozlišč ene komponente grafa nimajo nobenega vpliva na barvanje vozlišč drugih komponent. Pri problemu barvanja se bomo torej omejili na povezane grafe.

2.4 3-barvanja ravninskih grafov

Najbolj znan problem barvanja grafov je nedvomno izrek štirih barv, ki pravi, da je mogoče vsako lice ravninskega grafa pobarvati z eno izmed štirih barv tako, da noben par dveh priležnih lic ni enake barve. Z njim je bilo dokazano 4-barvanje ravninskih grafov. Kaj pa 3-barvanje? Ker je 3-barvanje poljubnih ravninskih grafov NP-težek problem, se je pojavilo vprašanje, ali obstajajo skupine ravninskih grafov, ki so vedno 3-obarvljive.

Grötzschev izrek [7] pravi, da je vsak ravninski graf brez ciklov dolžine 3 mogoče pobarvati s tremi barvami. Kasneje je Steinberg domneval, da je vsak ravninski graf brez ciklov dolžine 4 ali 5 mogoče pobarvati s tremi barvami. Steinbergova domneva je bila ovržena [5], porodila pa je vprašanje, kolikšno je najmanjše naravno število k, za katero velja, da obstaja 3-barvanje vseh ravninskih grafov brez ciklov dolžin 4 do k. Najmanjši do sedaj dokazan tovrstni k je 7 [4].

Mi smo se omejili na $k = 9$. Množico ravninskih grafov brez ciklov dolžin med 4 in 9 označimo z $\mathcal{G}_{4,9}$. Lica dolžine 3 imenujemo *trikotniki* ali *kratka lica*, lica, daljša od 3, pa *dolga lica*. Kljub temu, da se trikotniki lahko pojavijo v grafu, nikoli ne smejo biti priležni eden drugemu, saj bi tako nastal cikel dolžine 4. To pomeni, da imata lahko poljubna dva trikotnika le eno skupno vozlišče.



Slika 2.3: Pravilna postavitev trikotnikov (zgoraj) in napačna postavitev (spodaj).

Dokaz o 3-barvanju takšnih grafov se da enostavno prevesti v kvadratičen algoritem. Cilj te naloge je, da ta algoritem pohitrimo v linearni algoritem. V nadaljevanju bomo natančno opisali metodo prenosa naboja, ki je temelj dokaza obstoja barvanja in našega algoritma. Opisali bomo lica, ki se lahko pojavijo v grafih $\mathcal{G}_{4,9}$ in utemeljili, na katera od njih je treba paziti in zakaj. Pokazali bomo, da se glavni oviri, zaradi katere je algoritem kvadratičen, lahko izognemo tako, da se ne omejimo na barvanje 2-povezanih grafov.

Poglavje 3

Metoda prenosa naboja

Metoda prenosa naboja je postopek za delo z ravninskimi grafi. Strukturam v grafu — tipično vozliščem in licem — priredimo začetni naboj. V postopku prenosa naboje po grafu prestavljamo z uporabo izbranih pravil, pri tem pa ohranjamo skupno količino naboja. Tako dobljene končne naboje uporabimo za izpeljavo kombinatoričnih trditev o samem grafu.

Tipično je kumulativen začetni naboj negativen. Če postopke prenosa naboj ohranja, imamo tudi na koncu kakšno negativno nabito strukturo. V naši nalogi metodo prenosa naboja uporabimo za iskanje takšnih negativnih struktur (vozlišč, lic), ker jih lahko pobarvamo, ne glede na barve njihovih sosedov.

Pri ravninskih grafih metoda deluje tako: vsakemu vozlišču v dodelimo nek začetni naboj $c_0(v)$ glede na njegovo stopnjo in vsakemu licu f dodelimo nek začetni naboj $c_0(f)$ glede na njegovo dolžino. Uporabili smo naslednji dve pravili:

- vsako vozlišče v dobi začetni naboj $c_0(v) = d(v) - 6$,
- vsako lice f dobi začetni naboj $c_0(f) = 2\ell(f) - 6$.

$d(v)$ oz. $\ell(f)$	0	1	2	3	4	5	6	7	8	9	10	11	12	...	d
$c_0(v)$	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	...	d - 6
$c_0(f)$	/	-4	-2	0	2	4	6	8	10	12	14	16	18	...	$2l$ - 6

Tabela 3.1: Vozlišča, lica in njihovi naboji.

Ker vsaka povezava povezuje natanko dve vozlišči, je vsota stopenj vozlišč dvakrat večja od števila povezav. Prav tako, ker vsaka povezava meji na natanko dve lici, je vsota dolžin lic dvakrat večja od števila povezav.

$$\sum_{v \in V(G)} d(v) = 2|E|$$

$$\sum_{f \in V(F)} \ell(f) = 2|E|$$

Če vstavimo te podatke v šestkratnik Eulerjeve formule ((2.1)), dobimo

$$\sum_{v \in V(G)} (d(v) - 6) + \sum_{f \in F(G)} (2\ell(f) - 6) = -12, \tag{3.1}$$

kar pomeni, da je skupni seštevek nabojev vseh vozlišč in lic -12. Sledi, da so vsaj nekatera vozlišča ali lica negativno nabita.

Nato njihov naboj porazdelimo tako, da skupen naboj ostane isti. Naboje prenašamo z dolgih lic na prileżna vozlišča stopenj 3, 4 in 5 na naslednji način:

1. Vozlišče stopnje 3 prejme 1 naboj z vsakega prileżnega dolgega lica, če nobeno od teh treh lic ni trikotnik.

2. Vozlišče stopnje 3 prejme 3/2 naboja z vsakega prileżnega dolgega lica, če je eno od prileżnih lic kratko. Vozlišče stopnje 3 ne more imeti dveh prileżnih trikotnikov (zaradi omejitev v prejšnjem poglavju).

3. Vozlišče stopnje 4, ki ima dva prileżna trikotnika, prejme po 1 naboj z vsakega prileżnega dolgega lica.

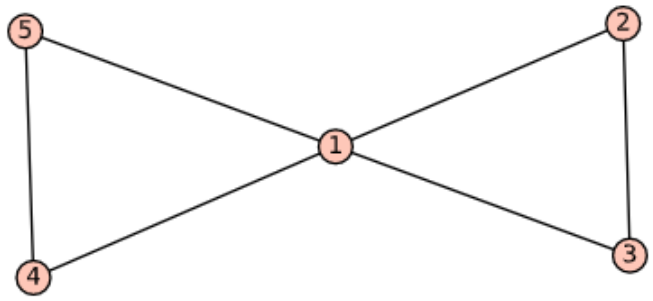
4. Vozlišče stopnje 4, ki ima samo en prileżen trikotnik, prejme 1 naboj z dolgega lica, ki je nasproti trikotnika in $\frac{1}{2}$ naboja z dolgih lic prileżnih trikotniku.

5. Vozlišče stopnje 4 brez priležnih trikotnikov prejme $\frac{1}{2}$ naboja z vsakega priležnega lica.

6. Vozlišče stopnje 5 prejme $\frac{1}{2}$ naboja z vsakega priležnega dolgega lica. Ta so vsaj tri, zaradi omejitve trikotnikov.

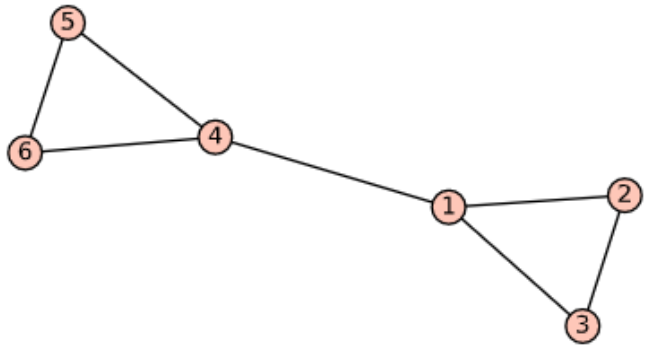
Četudi se v grafih $G \in \mathcal{G}_{4,9}$ ne morejo pojaviti cikli dolžin 4 do 9, to še ne pomeni, da grafi ne vsebujejo lic teh dolžin. Pojavijo se lahko lica dolžin 4, 5 in 7, vendar če iz grafa odstranimo veje (njihovo 3-barvanje je trivialno in možno, ne glede na barve sosedov veje), lica teh dolžin izginejo. Pazljivi moramo biti samo pri licih dolžin 6, 8 in 9, ki imajo začetni naboj 6, 10, oz. 12. Pojavi se vprašanje, ali je mogoče, da s temi pravili lica dolžin 6, 8 in 9 oddajo več naboja, kot ga na začetku dobijo? V naslednjih odstavkih bomo utemeljili, zakaj to ni mogoče.

Lice dolžine 6, sestavljeno iz dveh trikotnikov (prikazano na sliki 3.1), lahko sredinskemu vozlišču pošlje največ 1 naboj, ker je to vozlišče stopnje vsaj 4. To stori dvakrat, saj se vozlišče na licu pojavi dvakrat. Tudi ostalim vozliščem lahko pošlje največ 1 naboj. V primeru, da bi lahko poslalo 3/2 naboja, bi pomenilo, da je znotraj trikotnika še nov priležen trikotnik, kar pa ne ustreza našim kriterijem. Lice dolžine 6, z začetnim nabojem 6, lahko torej pošlje največ 6 naboja svojim vozliščem in zato ni negativno nabito.



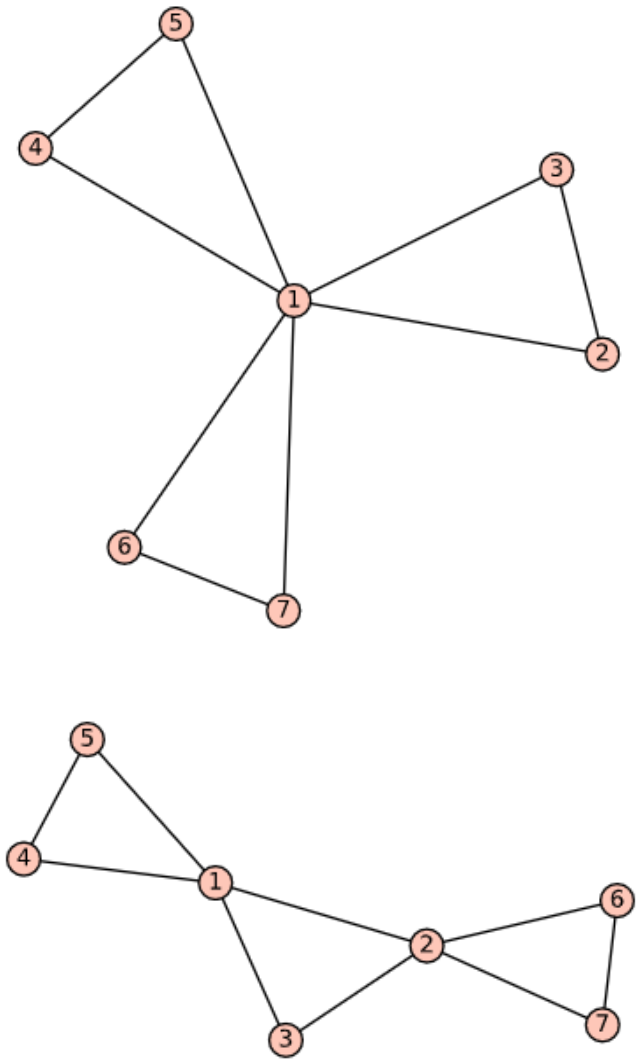
Slika 3.1: Zunanje lice je dolžine 6, a ni cikel.

Lice dolžine 8, ki ni cikel, je prikazano na sliki 3.2. Svojima notranjima vozliščema (na sliki sta to vozlišči 1 in 4) lahko pošlje največ 3/2 naboja, vsakemu dvakrat, ker se ti dve vozlišči na licu pojavita dvakrat. Zunanjim štirim pa lahko, zaradi istega pravila, kot pri licu dolžine 6, pošlje le 1 naboj vsakemu. Tako lahko torej lice dolžine 8, z začetnim nabojem 10, svojim vozliščem pošlje največ 10 naboja. S tem lice ostane nenegativno nabito.



Slika 3.2: Zunanje lice je dolžine 8, a ni cikel.

Lice dolžine 9 ima dve obliki (prikazani na sliki 3.3). Ker ni možno, da bi se pojavila dva prileżna trikotnika, lahko lice dolžine 9 vsakemu kotnemu vozlišču pošlje le 1 naboj. Ostala vozlišča pa so stopenj 4 ali več. Tudi tem vozliščem lahko pošlje vsakemu le 1 naboj. Tako lahko torej lice dolžine 9, z začetnim nabojem 12, svojim vozliščem pošlje največ 9 naboja. S tem tudi ta lica ostanejo nenegativno nabita.



Slika 3.3: Zunanji lici dolžine 9, a nista cikla.

To pomeni, da lica, krajša od 10, ne morejo biti negativno nabita. Kaj pa daljša lica? Ker lice lahko odda največ $3/2$ naboja vsakemu priležnemu vozlišču, začetni naboj lic pa je $2\ell(f)-6$, ima lice dolžine 12 vedno nenegativen naboj:

$$c_0(f_{12}) = 2 * 12 - 6 = 18,$$

$$c_1(f_{12}) = 18 - 12 * 3/2 = 0.$$

Sledi, da imajo lica, daljša od 12, lahko le pozitiven naboj. Ostaneta le lici dolžin 10 in 11. Da je lice dolžine 11 negativno, mora oddati vsakemu priležnemu vozlišču $3/2$ naboja. Da se to zgodi, mora vsako vozlišče izpolnjevati naslednja pogoja:

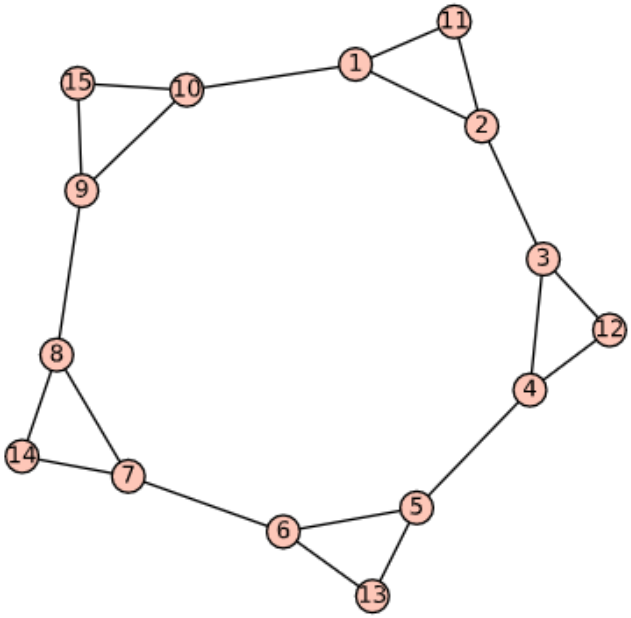
- vozlišče mora biti stopnje 3,
- vozlišče mora biti priležno natanko enemu trikotniku.

Prvi pogoj sam po sebi ni problematičen. Ker pa je lahko vsako vozlišče priležno le enemu trikotniku, imamo lahko le sodo število volišč na licu, ki izpolnjujejo oba pogoja. Zadnje vozlišče bodisi ni stopnje 3 ali pa ni priležno trikotniku. Torej vsaj enemu vozlišču to lice ne pošlje $3/2$ naboja, vendar največ 1, kar nas pripelje do:

$$c_0(f_{11}) = 2 * 11 - 6 = 16,$$

$$c_1(f_{11}) = 16 - (10 * 3/2 + 1) = 0.$$

Torej tudi lica dolžine 11 niso negativno nabita. Spomnimo pa se, da je skupni naboj vseh lic in vozlišč -12. Iz tega sledi, da v povezanih ravninskih grafih, brez vozlišč stopenj 0, 1 ali 2, nujno obstajajo lica dolžine 10, ki so negativno nabita. Izkaže se, da taka lica obstajajo in imajo prav določeno obliko.



Slika 3.4: Negativno nabito lice dolžine 10.

Takšno lice vsakemu svojemu vozlišču pošlje 3/2 naboja in s tem lice pride na -1 naboja. Ker je skupni naboj -12, sledi, da se taka struktura, v nadaljevanju imenovana 10-*zvezda* (ali pa samo *zvezda*), pojavi vsaj dvanajstkrat v grafu, če ta graf nima vozlišč stopenj 1 in 2. To strukturo pa, tako kot veje, znamo pobarvati, ne glede na barvo sosednjih vozlišč. Več o tem v razdelku 4.4.

To nas pripelje do naslednje razporeditve končnih ($c_1(v)$ oz. $c_1(f)$) nabojev:

$d(v)$ oz. $\ell(f)$	0	1	2	3	4	5	6	7	8	9	10	11	12
$c_1(v)$	-6	-5	-4	0	0	$\geqslant 0$	0	1	2	3	4	5	6
$c_1(f)$	/	/	/	0	/	/	$\geqslant 0$	/	$\geqslant 0$	$\geqslant 0$	$\geqslant -1$	$\geqslant 0$	$\geqslant 0$

Tabela 3.2: Naboji vozlišč in lic po prenosu naboja.

Kot je razvidno iz tabele 3.2, so negativno nabite strukture lahko le vozlišča stopenj 0, 1 in 2 in lica dolžine 10. Pri problemu barvanja so vozlišča stopnje 0 in 1 trivialna, zato jih lahko preprosto odmislimo (oz. odstranimo). Enako velja za veje (vozlišča stopnje 2). Ne glede na barve sosedov končnih vozlišč veje, lahko vejo pobravamo. Na tak način se znebimo vseh negativno nabitih vozlišč. Če po tem, ko v grafu odstranimo vsa vozlišča stopnje 0, 1 in 2, graf ni prazen, to pomeni, da mora obstajati vsaj eno negativno nabito lice, ker je skupni naboj vseh lic in vozlišč še vedno -12. Dokazali pa smo že, da lice, ki bi imelo naboj manj od -1, ne obstaja. Obstaja torej vsaj 12 negativno nabitih lic.

Poglavje 4

Algoritem

Cilj naloge je bil razviti algoritem, ki pobarva grafe iz $\mathcal{G}_{4,9}$ z le tremi barvami in sicer v linearnem času. Relativno enostavna posledica Eulerjeve formule (2.1) je, da imajo ravninski grafi število povezav omejeno z linearno funkcijo števila vozlišč. Zato lahko kot velikost vhoda izberemo kar število vozlišč n. Vsako iteracijo po vseh vozliščih, po vseh povezavah ali po vseh licih torej lahko opravimo v $O(n)$ časa.

4.1 Prerezna vozlišča

Naš algoritem je zasnovan na podlagi postopka za barvanje grafov brez kratkih ciklov, ki ga je predstavil Salavatipour [10]. Ta postopek lahko prevedemo v algoritem, ki grafe iz $\mathcal{G}_{4,9}$ pobarva v linearnem času. Vendar barva le grafe, ki so 2-povezani. V primeru, ko graf ni 2-povezan, ima prerezno vozlišče. Algoritem na vsakem koraku preveri, ali graf G vsebuje tako vozlišče. Če graf G vsebuje prerezno vozlišče x, razdeli G na fragmenta G_1 in G_2 glede na x tako, da velja:

$$G = G_1 \cup G_2 \quad \text{in} \quad G_1 \cap G_2 = x.$$

Nato algoritem vsak fragment G_1, G_2 posebej rekurzivno pobarva. Ker si G_1 in G_2 delita vozlišče x, je potrebno doseči, da se barvanji G_1 in G_2 ujemata

v x. To naredimo tako, da vse barve vozlišč grafa G_2 popravimo z isto konstanto. Časovna zahtevnost popravljanja barv vozlišč G_2 pa je linearno odvisna od velikosti G_2. Časovni problem se pojavi v primerih, ko je graf G_2 podobne velikosti kot vhodni graf G, torej ko graf G_1 vsebuje minimalno število vozlišč. Če na vsakem koraku graf G na n vozliščih razcepimo na graf G_1 konstantne velikosti c (kjer je $c \ll n$) in graf G_2 velikosti $n-c$, je časovna zahtevnost popravljanja enaka

$$\Theta(n-c) + \Theta(n-2c) + \Theta(n-3c) + \cdots = \Theta(n^2),$$

kar je kvadratične časovne zahtevnosti.

Algoritem 1

Vhod: graf G, končni naboji $c(G)$ grafa G
function OBARVAJ(G, $c(G)$)
  if G vsebuje prerezno vozlišče x **then**
    $b_1 = obarvaj(G_1, c(G_1))$
    $b_2 = obarvaj(G_2, c(G_2))$
    $b_2' = b_2 + (b_1(x) - b_2(x))$
    return $(b_1 \cup b_2')$
  else
    poišči negativno nabito strukturo x
    določi popravljene naboje $c(G-x)$
    $b = obarvaj(G-x, c(G-x))$
    b' = razširitev b na x
    return b'
  end if
end function

Zaradi tega smo se pri iskanju linearnega algoritma osredotočili na prerezna vozlišča. Salavatipour [10] se je omejil na barvanje 2-povezanih grafov, ker imajo takšni grafi lastnost, da je vsak cikel hkrati tudi lice. Če torej graf nima

cikla dolžine 6, tudi nima lica dolžine 6. Mi smo pa v poglavju 3 pokazali, da tudi če dopuščamo lica dolžin 4 do 9, bodo taka lica vedno nenegativno nabita. S tem lahko opustimo zahtevo po 2-povezanosti in se osredotočimo na barvanje grafov s prereznimi točkami. To nam algoritem posploši in pohitri.

Algoritem 2 Popravljeni algoritem

Vhod: graf G, končni naboji $c(G)$ grafa G
function OBARVAJ(G, $c(G)$)
    poišči negativno nabito strukturo x
    določi popravljene naboje $c(G - x)$
    $b = obarvaj(G - x, c(G - x))$
    b' = razširitev b na x
    return b'
end function

4.2 Priprava podatkov

Sama priprava podatkov je odvisna od formata, v katerem so podani podatki grafa. V našem primeru iz generatorja dobimo seznam vozlišč z urejenim seznamom sosedov vsakega vozlišča, seznam povezav in seznam lic z urejenim seznamom povezav, ki to sestavljajo lice. Naš algoritem potrebuje le seznam vozlišč s sosedi in seznam lic z vozlišči (ali povezavami). V primeru, da algoritem prejme le seznam vozlišč in povezav, morajo biti sosedi nujno urejeni (smer orientacije ni pomembna, pomembno je le, da je povsod enaka), ker so le tako lica enolično določena.

Z iteracijo po seznamu vozlišč se naredi tabela za razred *vozlišče*, kjer se za vsako vozlišče zapiše unikatni identifikator (ki je nastavljen kar na indeks vozlišča v tabeli), seznam sosedov vozlišča, izračuna se stopnja, doda prazen seznam prileţnih lic, pobarva vozlišče na *nepobarvano* in naredi prazna polja za še druge informacije, ki v tistem trenutku v algoritmu še niso znane.

Druga tabela je tabela z elementi razreda *lice*, kjer se za vsako lice zapiše identifikator lica, iz seznama prileţnih povezav razbere dolţina lica in katera vozlišča ga sestavljajo. V tem koraku se tudi zapiše identifikator lica v seznam prileţnih lic teh vozlišč. Prav tako kot pri vozliščih, bo kasneje vsako lice dobilo še nove atribute.

Ker se vsako vozlišče pojavi le v toliko licih, kot je stopnja vozlišča, povprečna stopnja vozlišč v ravninskih grafih pa je strogo manjša od 6, se bo vsako vozlišče v povprečju pojavilo v manj kot šestih licih, kar ohranja linearno časovno zahtevnost.

Naslednji korak je začetni naboj. Algoritem iterira čez vsako vozlišče v in lice f v tabelah in mu dodeli nov atribut z imenom *naboj* po že prej omenjeni formuli $d(v) - 6$, vsakemu licu f pa naboj $2\ell(f) - 6$. Temu sledi prerazporejanje naboja. Pogleda se stopnja vsakega vozlišča in dolţine vseh njegovih prileţnih lic, odšteje primeren naboj vsakemu licu in se ga prišteje vozlišču.

Zadnji korak pri pripravi podatkov je izgradnja *tabele nabojev*, kamor zapišemo vsa vozlišča in lica. Glede na njihov naboj jih razvrstimo v eno izmed šestih *skupin*:

- vozlišča z nabojem -6 (torej stopnje 0),
- vozlišča z nabojem -5 (stopnje 1),
- vozlišča z nabojem -4 (stopnje 2),
- vozlišča z nenegativnim nabojem,
- lica z nabojem -1 (zvezde dolţine 10),
- lica z nenegativnim nabojem.

Vsako vozlišče in lice zapišemo v eno skupino in hkrati vozlišču oz. licu dodamo nov atribut: *dvojico skupina/pozicija* vozlišča oz. lica v tej skupini. To nam omogoča dostop v konstantnem času.

4.3 Brisanje in posodabljanje podatkov

Po končani pripravi podatkov se v algoritmu začne brisanje vozlišč. Od seznamov, predstavljenih v poglavju 4.2, je urejenost pomembna le pri seznamu sosedov vozlišč. Urejenost ostalih seznamov ni pomembna, zato lahko brisanje elementov iz teh tabel izvedemo v konstantnem času. To storimo tako, da element, ki ga želimo izbrisati, preprosto zamenjamo z zadnjim elementom seznama, nato pa zadnji element seznama izbrišemo. To pomeni, da se takoj, ko prvi element zbrišemo, vrstni red elementov v seznamu spremeni. Da ohranimo dostop do poljubnega elementa, vsakemu vozlišču in licu pripišemo nove atribute, ki nam povedo, kje v *tabeli nabojev* se nahajajo.

Vsakič, ko odstranimo negativno nabito vozlišče v (ali lice) iz grafa G, dobimo novi graf $G' = G - v$. Ta postopek ponavljamo, dokler ne dobimo grafa na enem samem vozlišču. Ampak ker je G' zelo podoben grafu G, je večina nabojev enakih. Razlika je le v nabojih sosedov in prileţnih lic odstranjene strukture v. Pri vsakem brisanju vozlišča (ali lica) na novo izračunamo naboje samo tem sosedom in prileţnim licem.

Brisanje vozlišč/lic razdelimo na štiri kategorije: brisanje vozlišč stopnje 0, brisanje vozlišč stopnje 1, brisanje vozlišč stopnje 2 in brisanje zvezd dolžine 10. Glavna razlika med kategorijami je obnašanje prileţnih lic pri brisanju.

- Pri brisanju vozlišča stopnje 0 je stanje lic nepomembno.
- Pri brisanju vozlišča stopnje 1 se število lic ohrani, edinemu prileţnemu licu pa se dolžina zmanjša za 2.
- Pri brisanju vozlišča stopnje 2 se število vseh lic v preostalem grafu zniža za 1, če to vozlišče ni prerezno. Sicer se število lic poveča za 1.
- Pri brisanju zvezde se enajst lic (eno lice dolžine 10, pet prileţnih trikotnikov in pet prileţnih dolgih lic) združi v eno samo, če ta zvezda ni prerezna. Sicer iz obstoječih 7 do 10 lic nastane 5 do 2 novih lic (odvisno od števila komponent, na katere graf razpade).

Brisanje vozlišča stopnje 0 je trivialno. Pri nobeni drugi strukturi se naboji ne spremenijo. Vozlišče preprosto odstranimo iz tabele nabojev in dodamo na vrh *sklada odstranjenih struktur* in nadaljujemo z naslednjim vozliščem.

V primeru, ko je brisani element vozlišče stopnje 1, ga dodamo na vrh sklada odstranjenih struktur, ter izbrišemo iz tabele nabojev in s seznama sosedov pri sosednjem vozlišču. Temu sosedu posodobimo tudi naboj. Ker se je dolžina prileznega lica zmanjšala, preverimo dolžino. To storimo tako, da začnemo pri sosedu brisanega vozlišča in se "sprehodimo" po povezavah lica. Pri tem je zelo pomembno, da je seznam sosedov vozlišč urejen, sicer bi to ne bilo možno. Če v desetih korakih ne pridemo nazaj do začetnega vozlišča, pomeni, da je lice še vedno daljše od 10 in ima zato nenegativen naboj. V primeru, da je lice po novem dolžine 10, mu izračunamo naboj in ga po potrebi prestavimo v drugo skupino v tabeli nabojev. Zapišemo si tudi, skozi katerih deset vozlišč smo se sprehodili. Ker se po desetih korakih ustavimo, je vsako tako preverjanje dolžine lica opravljeno v konstantnem času.

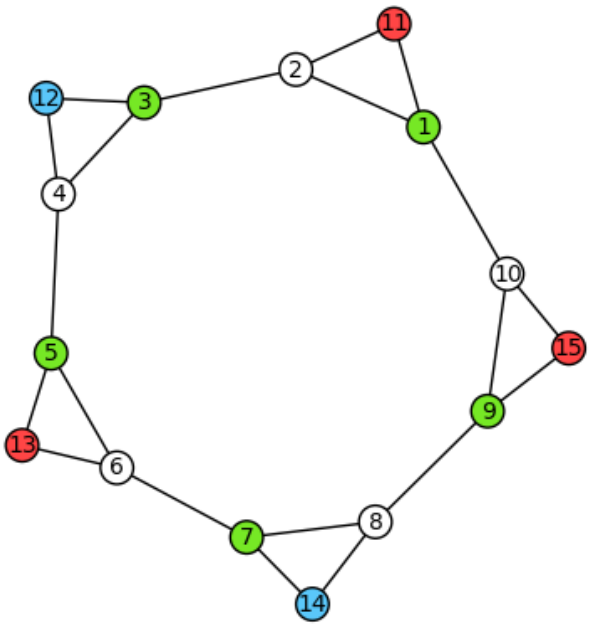
Prvi problem se pojavi šele pri brisanju vozlišča stopnje 2. Ne vemo, ali je to vozlišče prerezno ali ne, zato tudi ne vemo, kolikšno bo skupno število lic po brisanju, a to ni pomembno. Pri obeh sosedih brisanega vozlišča začnemo sprehod po desetih povezavah v licu. Če pri katerem od sprehodov v desetih korakih pridemo nazaj do začetnega vozlišča, je to novo lice dolžine 10 in ga dodamo v tabelo nabojev. Sosedoma brisanega vozlišča posodobimo seznam vozlišč in naboj. Po potrebi ju prestavimo na drugo mesto v tabeli nabojev.

Zadnji tip struktur, ki jih brišemo, je zvezda. Brisanje zvezde je zelo podobno brisanju vozlišča stopnje 2. Vsaka zvezda ima pet sosednjih vozlišč (v razdelku 4.4 jih imenujemo *sosedi zvezde*), na katere je pripeta. Ko zvezdo izbrišemo in jo dodamo na sklad odstranjenih struktur, si moramo njene sosede zapomniti. Po brisanju pri vsakem od petih sosedov zvezde začnemo nov sprehod dolžine 10, da ugotovimo, ali smo dobili kakšno novo lice dolžine 10. Ne smemo pozabiti na posodobitev nabojev sosednjih vozlišč.

4.4 Barvanje

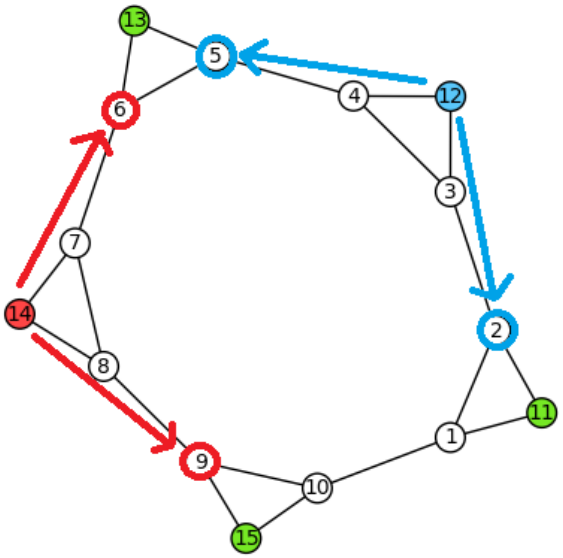
Na koncu algoritmu ostane še barvanje vozlišč. Elemente jemlje s *sklada odstranjenih struktur*, enega za drugim in jih pobarva. Če je dobljeni element vozlišče, algoritem preveri barve njegovih že pobarvanih sosedov. Vozlišče nato pobarva z ustrezno barvo. V primeru, ko je element na vrhu sklada odstranjenih struktur 10-zvezda, je treba biti bolj previden. Potrebno je definirati *sosede zvezde*. Za sosede zvezde velja tistih pet vozlišč (na slikah 4.1 in 4.2 so to vozlišča 11, 12, 13, 14 in 15), ki jih dobimo, če vzamemo vse sosede vozlišč zvezde in nato odstranimo vozlišča, ki sestavljajo centralno lice zvezde. Ločimo dva primera:

1. Sosedi zvezde so pobarvani z eno ali dvema barvama. V tem primeru vsako drugo vozlišče zvezde pobarvamo z neuporabljeno barvo (na sliki 4.1 zeleno). Barve ostalih petih vozlišč so s tem določene.



Slika 4.1: Barvanje zvezde, kjer so njeni sosedi največ dveh različnih barv.

2. Sosedi zvezde so pobarvani s tremi barvami. V tem primeru začnemo barvanje tako, da pri vsakem sosedu zvezde, ki je unikatne barve, pobarvamo najbližji dve vozlišči, ki mu nista prileżni, z barvo tega soseda. Primer je prikazan na sliki 4.2. Ker je vozlišče 12 drugačne barve od vozlišč 11, 13, 14 in 15, vozlišči 2 in 5 pobarvamo z barvo vozlišča 12 (modro). Enako storimo, če obstaja še drug sosed unikatne barve (v tem primeru je to vozlišče 14).



Slika 4.2: Barvanje zvezde, kjer so sosedi treh različnih barv.

Ta postopek nadaljujemo, dokler sklad odstranjenih struktur ni prazen. Če smo vse izvedli po zgoraj opisanem algoritmu, je graf pravilno 3-pobarvan.

Poglavje 5

Testiranje

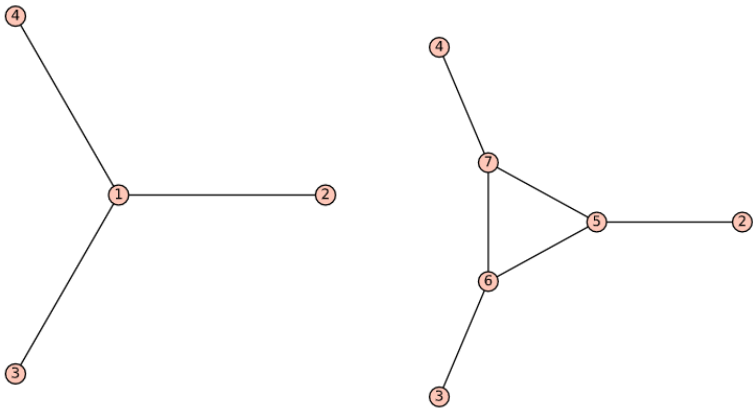
Algoritem, predstavljen v prejšnjem poglavju, smo preizkusili na množici testnih grafov. V tem poglavju najprej razložimo, kako testne grafe generiramo.

V poglavju 3 smo ugotovili, da vsak graf $G \in \mathcal{G}_{4,9}$, ki nima vozlišč stopenj nič, ena ali dva, vsebuje vsaj dvanajst 10-zvezd (slika 3.4). Enake zvezdne strukture dobimo, če uporabimo operacijo *prisekanje vozlišča* na vozliščih dodekaedra.

Definicija 5.1 *Prisekanje vozlišča v je operacija na ravninskem grafu, kjer vozlišče v stopnje d zamenjamo z licem dolžine d. To novo lice ima d vozlišč, vseh stopnje* 3*. Vsako je povezano z dvema drugima vozliščema v novem licu.*

Prav tako dobimo željenih 12 zvezd s prisekanjem vozlišč katerega koli fulerena, ne samo dodekahedrona. Za generiranje fulerenov smo uporabili generator buckygen [8].

Definicija 5.2 *Fuleren je kubičen,* 3*-povezan, ravninski graf, ki vsebuje samo lica dolžin* 5 *in* 6*. Vsak fuleren vsebuje natanko* 12 *ciklov (in tako tudi lic) dolžine* 5*.*



Slika 5.1: Prisekanje vozlišča stopnje 3.

Najmanjši fuleren je zgoraj omenjeni dodekaeder, ki ima le petkotnike in nobenega šestkotnika. Najbolj znan fuleren se imenuje buckminster fuleren [9], ki ga večina pozna kot graf nogometne žoge.

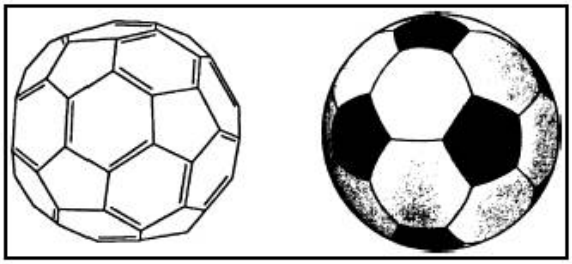


Fig. 2.13. C_{60} fullerene "buckyball" and a soccer ball.[382]

Slika 5.2: Fuleren in nogometna žoga, vzeto iz [6].

S prisekanjem fulerenov smo torej dobili ravninske grafe z lici dolžin 3 in 12 in z natanko 12 lici dolžine 10. Za testiranje smo morali razred testnih grafov še razširiti. Na generiranih grafih smo naključno izvajali naslednje operacije:

1. subdivizijo povezave (na povezavo vstavimo vozlišče stopnje 2),

2. dodajanje novih vozlišč, vej, trikotnikov in dolgih ciklov, ki smo jih nato pripeli na vozlišče že obstoječega grafa,

3. dodajanje tetiv preko že obstoječih lic, skupaj z njihovo večkratno subdivizijo, da so novonastala lica postala ustrezno dolga,

4. lepljenje dveh generiranih grafov:

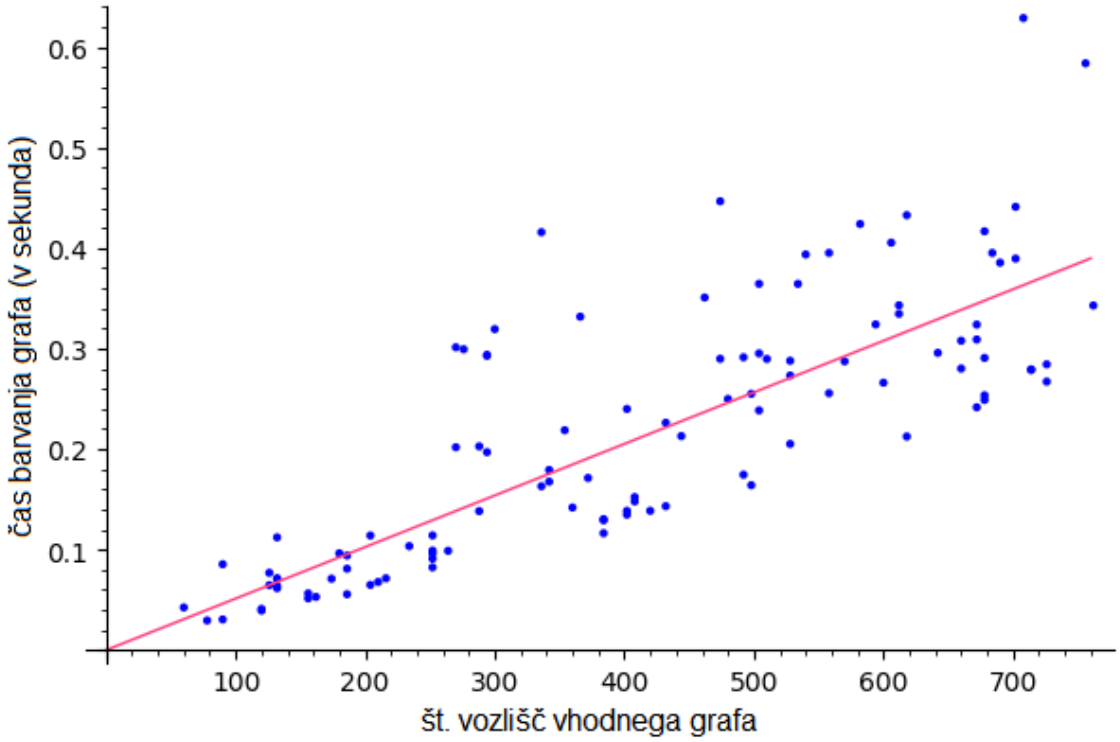
   - z dodajanjem vej med grafoma,
   - z identifikacijo lic iste dolžine.

Pazili smo, da nismo izvedli le enkratne subdivizije na povezavah trikotnikov, saj bi tako dobili cikle dolžine štiri. Potrebna je vsaj sedem ali večkratna subdivizija povezave na povezavah trikotnika, da ta trikotnik postane lice dolžine vsaj 10. Tudi pri lepljenju lic je bilo treba paziti, da bi ne dobili dveh sosednjih trikotnikov, kar bi nam dalo cikel dolžine štiri.

Na ta način smo dobili mnogo raznolikih grafov. Seveda obstajajo tudi grafi iz $\mathcal{G}_{4,9}$, ki jih naš generator ne generira. Vseeno pa je generirana družina zadosti velika, da smo lahko uspešno testirali algoritem.

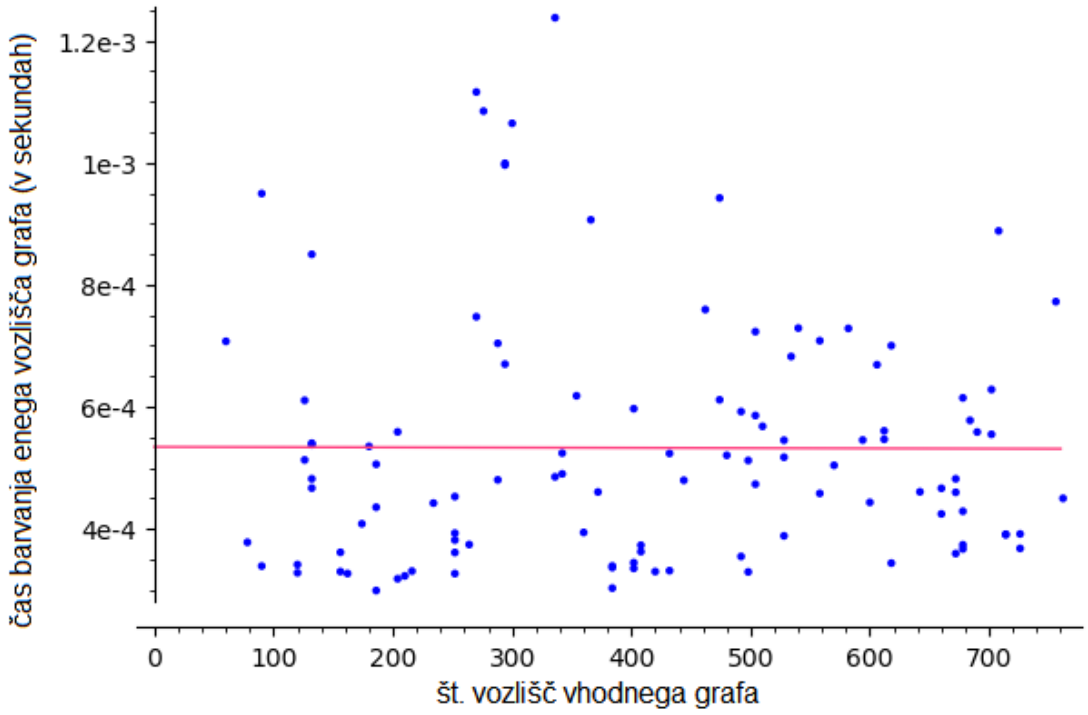
5.1 Rezultati testiranja

Merili smo čas, ki ga je algoritem potreboval za barvanje grafov različnih velikosti, kar je prikazano sliki 5.3.



Slika 5.3: Čas barvanja grafov (v sekundah) v odvisnosti od števila vozlišč posameznega vhodnega grafa in regresijska premica.

Iz grafa je empirično razvidno, da je čas barvanja linearno odvisen od števila vozlišč v grafu. Izračunali smo tudi povprečen čas barvanja posameznega vozlišča v odvisnosti od velikosti grafa, kar je prikazano na sliki 5.4.



Slika 5.4: Povprečen čas barvanja posameznega vozlišča v odvisnosti od števila vozlišč v vhodnem grafu.

Iz slik smo odstranili štiri primere barvanja. Čas, porabljen za barvanje teh štirih primerov, je bil mnogo (v enem primeru celo trikrat) daljši od časa barvanja njim podobnih grafov. Kljub temu so ta barvanja upoštevana v rezultatih. To odstopanje pripisujemo počasnejšemu delovanju računalnika v času barvanja, ne pa temu, da bi algoritem v teh primerih naredil več korakov.

Iz podatkov, prikazanih na slikah 5.3 in 5.4, smo z metodo najmanjših kvadratov izračunali enačbi premic, ki se našim rezultatom najbolj prilegata (prikazano z rdečo). Iz slike 5.4 je razvidno, da povprečen čas barvanja vozlišča ne narašča z velikostjo grafa.

Poglavje 6

Zaključek

Ob razmišljanju, kako se lotiti naloge, smo skušali predvideti morebitne zaplete. Vedeli smo, da bi algoritem teoretično sicer moral delovati. Skušali smo tudi predvideti ovire, na katere bi utegnili naleteti (generiranje grafov $\mathcal{G}_{4,9}$, lica, krajša od 10 in druge), zavedali pa smo se, da bo gotovo še kakšna, ki je nismo pričakovali (obnašanje lic pri brisanju prereznih vozlišč/lic).

Salavatipourjev algoritem, ki izkorišča prerezna vozlišča, ima teoretično kvadratično časovno zahtevnost. V delu smo pokazali, da je grafe iz $\mathcal{G}_{4,9}$ možno obarvati v linearnem času — in sicer s poenostavljenim algoritmom. Teoretično smo pokazali, da časovno potraten del Salavatipourjevega pristopa niti ni potreben. Algoritem smo implementirali in časovno ovrednotili. Praktična časovna zahtevnost res ustreza teoretični.

Literatura

[1] Kenneth Appel and Wolfgang Haken. Every planar map is four colorable. *Bulletin of the American mathematical Society*, 82(5):711–712, 1976.

[2] Oleg V Borodin. Structural properties of plane graphs without adjacent triangles and an application to 3-colorings. *Journal of Graph Theory*, 21(2):183–186, 1996.

[3] Oleg V Borodin, Alexei N Glebov, Mickaël Montassier, in André Raspaud. Planar graphs without 5-and 7-cycles and without adjacent triangles are 3-colorable. *Journal of Combinatorial Theory, Series B*, 99(4):668–673, 2009.

[4] Oleg V Borodin, Alexei N Glebov, André Raspaud, in Mohammad R Salavatipour. Planar graphs without cycles of length from 4 to 7 are 3-colorable. *Journal of Combinatorial Theory, Series B*, 93(2):303–311, 2005.

[5] Vincent Cohen-Addad, Michael Hebdige, Zhentao Li, Esteban Salgado, et al. Steinberg's conjecture is false. *Journal of Combinatorial Theory, Series B*, 122:452–456, 2017.

[6] Robert A Freitas. *Nanomedicine, volume I: basic capabilities*, volume 1. Landes Bioscience Georgetown, TX, 1999.

[7] H Grötzch. Zur theorie der diskreten gebilde. vii. ein dreifarbensatz für dreikreisfreie netze auf der kugel. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg, Math-Nat.(1958/1959)*, str. 109–120.

[8] Brendan McKay, Gunnar Brinkmann, Jan Goedgebeur. Buckygen fullerene generator. Dosegljivo: `https://caagt.ugent.be/buckygen/`, 2012.

[9] Harold W Kroto, James R Heath, Sean C O'Brien, Robert F Curl, and Richard E Smalley. C60: Buckminsterfullerene. *Nature*, 318(6042):162, 1985.

[10] Mohammad Reza Salavatipour. *Graph colouring via the discharging method*. University of Toronto, 2003.