Poročilo projekta - Voronoijevi diagrami

Anže Mramor in Peter Tiselj

4.12.2020

Poročilo projekta - Voronoijevi diagrami

Uvod - Definicija Vornoijevih diagramov

Voronoijeve diagrame definiramo na ravnini ali na splošnem metričnem prostoru. V danem grafu G in podmnožici vozlišč U ($U \subseteq G$) razdelimo vozlišča grafa G, glede na to, katero vozlišče iz U jim je najbližje. Bolj natančno: za $\forall u \in U$, definiramo Voronoijevo celico, (u, U), kot množico vseh vozlišč iz G, ki so bližje u kot kateremu koli drugemu vozlišu $U \setminus \{u\}$.\ Oziroma s formalno definicijo:\ Naj bo X metrični prostor z razdaljo d, ki je definirana kot razdalja med točko $x \in X$ in podmnožico $A : d(x, A) = inf\{d(x, A)|a \in A\}$. Naj bo še K množica indeksov in $(P_k)_{k \in K}$ terica (tuple) nepraznih podmnožic v X. Potem definiramo Voronoijevo celico kot R_k za $k \in K$ kot

$$R_k = \{x \in X | d(x, P_k) \leq d(x, P_j); \forall j \neq k\}$$

Voronoijev diagram je enostavno terica Voronoijevih celic $(R_k)_{k \in K}$

Opis problema

Pogledala sva si enostavna omrežja:
- dvo- in trodeminzionalne mreže različnih velikosti - $1 \times n$, $2 \times n$, $n \times n$, ...
- različne tipe grafov kot so binarna drevesa in grafi s cikli

Za vsako omrežje sva si izbrala več smiselnih vrednosti k, ki so nama predstavljali število naključno izbranih vozlišč, za katere sva izračunala Voronoijeve celice. Nato sva statistično obdelovala velikosti dobljenih Voronoijevih celic. Pogledala sva povprečno število vozlišč v minimalni, maksimalni in povprečni Voronoijevi celici za vsako omrežje in vse izbrane k, poskušala iz grafov z manjšim številom vozlišč napovedati kakšno bo število vozlišč v celicah pri grafih z večjim številom vozlišč, ter poskušala ugotoviti kakšna je porazdelitev velikosti celic glede na izbrani k.
Za vsak izbrani k sva program pognala večkrat, saj sva tako lahko opazovala kako se pri različnem izboru začetnih vozlišč, spreminjajo zgornje vrednosti. Rezultate sva shranjevala v *.tsv* datoteke in jih nato obdelala v programu *r*. Ker so tabele za najin problem zelo velike, bova rezultate predstavila predvsem v obliki grafov.

Glavna ideja programa / psevdokoda

Problema sva se lotila tako, da sva sprva spisala program za generiranje Voronoijevih celic iz grafa in šele nato kodo za generiranje želenih grafov, v nadaljevanju pa bo program predstavljen v bolj logičnem vrstnem redu.

Generiranje grafov

Končni cilj tega dela programa je vrniti matriko sosednosti za želen tip grafa. Za vse grafe sva po predlogu prof. Vidalija skonstruirala razred Graf, ter nato še poseben razred za vsak tip grafa posebej. Razredi omogočajo generiranje različnih matrik sosednosti za vsak tip grafa, ter poenostavijo genriranje Voronoijevih celic za vsak tip, saj omogočajo enostavno klicanje za vsak tip, ter različna števila vozlišč grafov.
Poleg tega vsakemu izmed razredov pripada posamezna funkcija, s pomočjo katere se skonstruira primerna

matrika sosednosti za dan tip grafa. Te funkcije sva skonstruirala za vsak tip grafa, glede na njihove zakonitosti npr. v 2D mrežah ima vsako vozlišče lahko največ 4 sosednja vozlišča, v binomskih drevesih pa največ 3. 3D mreže so skonstruirane na podoben način kot 2D mreže, le v treh dimenzijah, cikli pa se generirajo iz binomskih dreves z dodajanjem povezav. Natančnejša razlaga kako sva konstruirala posamezen tip grafa je na voljo v komentarjih ob kodi v programu samem.

Konstruiranje Voronoijevih celic za posamezen graf

Ko enkrat imamo program, ki nam skonstruira matriko sosednosti za poljuben tip grafa določen z dimenzijami ali številom vzolšič, lahko zgeneriramo Voronoijev diagram za ta graf. Za to najprej potrebujemo razdalje med vsemi vozlišči znotraj grafa. Za to sva uporabila znan Floyd-Warshallov algoritem, ki iz matrike sosednosti vrne najkrajše poti med vsemi pari vozlišč. Nato te vrednosti uporabi glavna funkcija programa, Voronoi, ki kot podatke vzame matriko sosednosti želenega grafa, množico U vseh središč Voronoijevih celic, ter tip grafa (da vemo za kateri tip grafa iščemo diagram), ter vrne seznam seznamov. V vsakem od notranjih seznamov je prvi element središče Voronoijeve celice, sledijo pa vsa ustrezna vozlišča.
Funkcija deluje tako, da za graf s pomočjo Floyd-Warshalla določi najkrajše razdalje vseh vozlišč grafa do izbranih središč celic. Nato po stolpcih dobljene matrike primerja vse razdalje med seboj in poišče najkrajšo, ki jo potem doda v ustrezen seznam (k pripadajočemu središču). Če je neko vozlišče enako oddaljeno od več različnih središč (razdalja med vozliščem in središči je enaka za dve središči ali več), ga program doda na seznam k vsem središčem. Funkcija nato zapiše končni seznam seznamov v obliki *data frame*-a.

Generiranje rezultatov

Na koncu sva sestavila funkcijo, ki nam generira Voronoijeve diagrame za določen tip in število vozlišč, ter jih zapisuje v tekstovno datoteko. Odločila sva se, da želiva preveriti diagrame za čim več različih vrednosti k, saj bi tako dobila najboljše primerjave za grafe. Vendar se je hitro izkazalo, da bo program precej časovno zahteven, zato sva se odločila, da si bova ogledala le vrednosti k med 5 in 60 odstotki vseh vozlišč grafa, saj sva tam pričakovala najbolj zanimivo dogajanje. Za recimo grafe s 100 vozlišči sva torej vzela $k = 5, 6, 7, ..., 60$. Odločila sva se tudi, da bi bilo smiselno generiranje za vsak graf večkrat ponoviti. Ker je bilo spet očitno, da bo za velike grafe to lahko časnovno zelo dolgotrajno, sva se odločila, da bova za grafe z največ 100 vozlišči generirala 50 vzorcev, za večje pa le 20. Za primerjavo - za konstruiranje diagrama s 100 vozlišči je program potreboval povprečno 22 sekund, za 200 vozlišč povprečno 36 sekund, za 500 vozlišč pa že približno 25 minut, iz česar lahko razberemo, da je časovna zahtevnost programa zelo hitro naraščajoča, verjetno eksponentna.
Vse rezultate za en tip grafa s stalnim številom vozlišč sva shranila v isto tekstovno datoteko, ter na koncu programa dodala še funkcijo main, s katero sva lahko generirala diagrame za več različnih tipov in števil vozčišč hkrati.

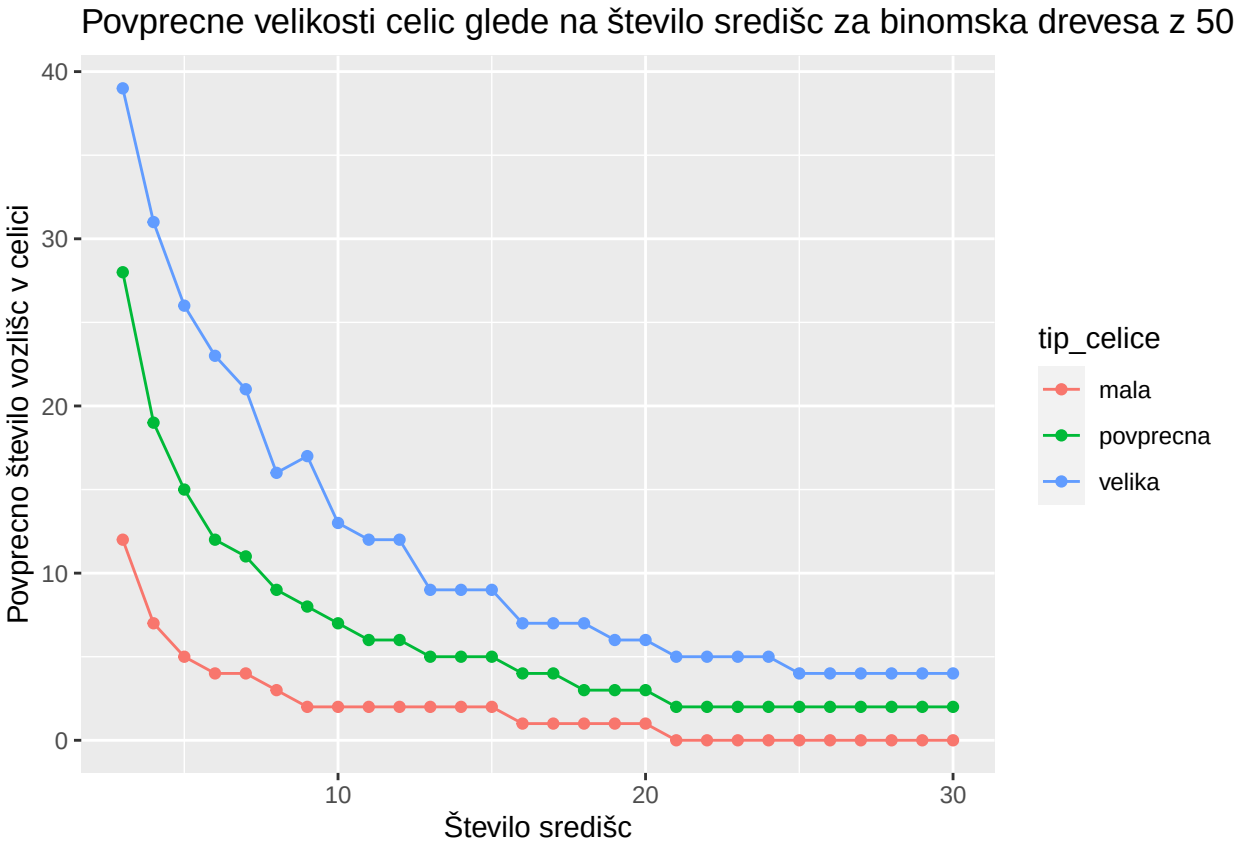
Statistično modeliranje v programu r

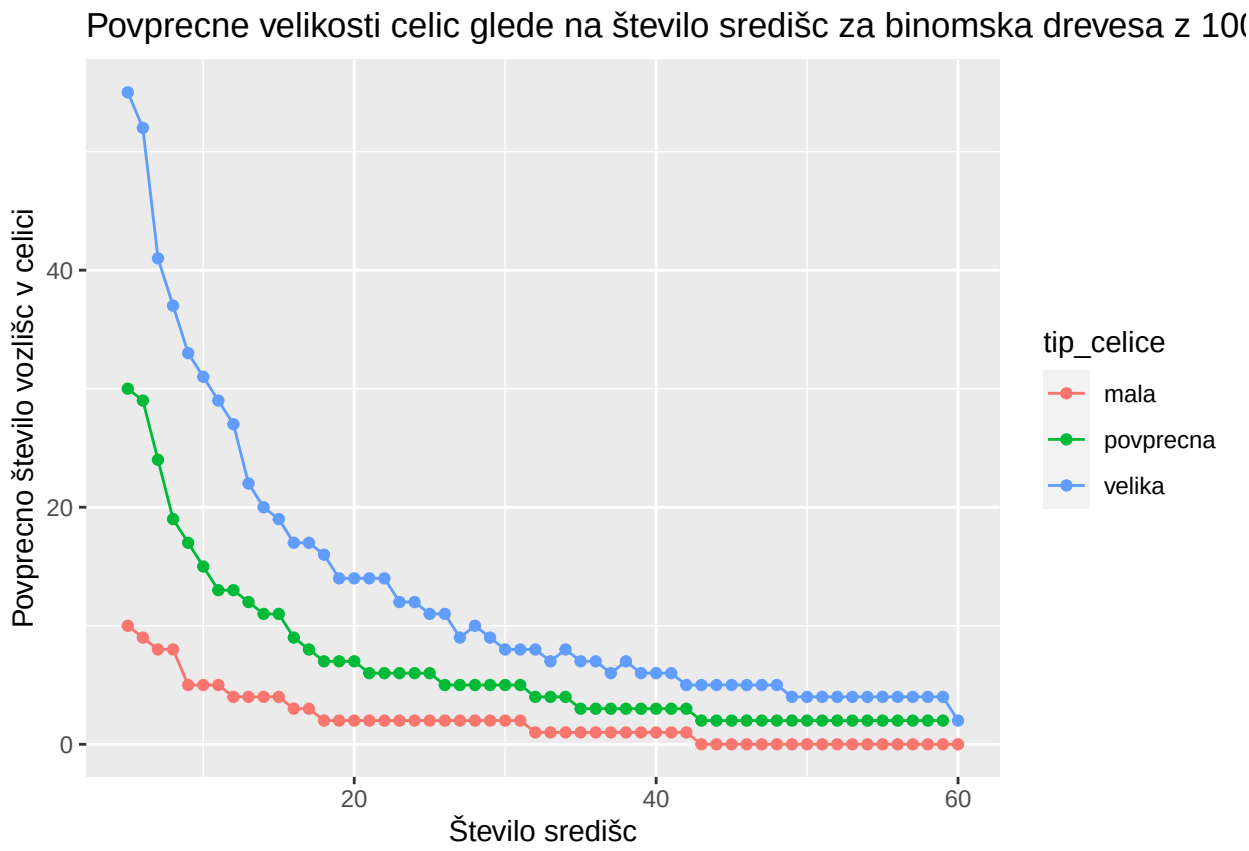
Obdelava podatkov v programu r

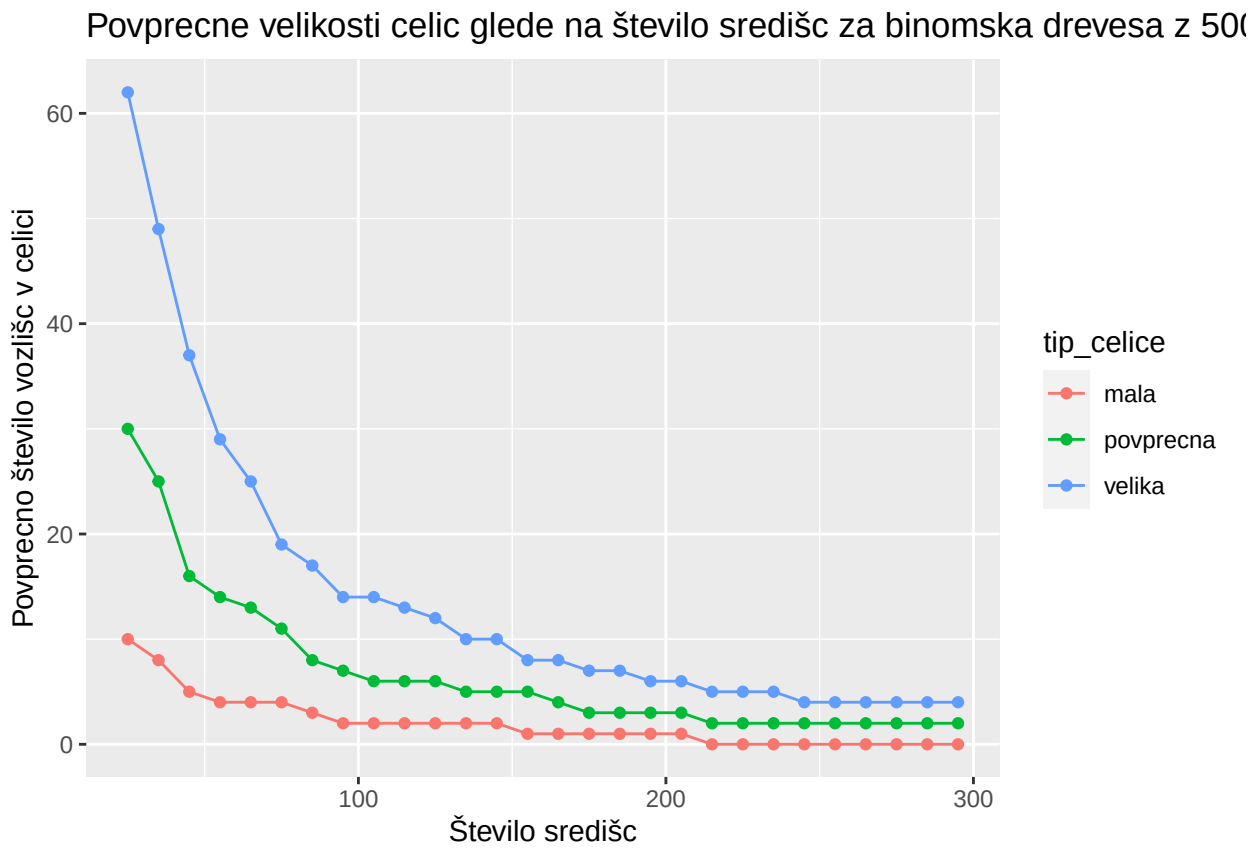
Podakte sva uvozila v program r in jih statistično obdelala. Najprej sva spisala funkcijo, ki je vsaki celici priredila status velike, povprečne ali male, glede na število elementov v njej. Najprej sva preračunala povprečno velikost vseh celic, glede na število središč. Kot velike celice sva nato nastavila tiste, ki so bile od povprečnih večje za faktor 1.9, kot male pa tiste, ki so bile manjše za faktor 0.1. Nato sva za vsak tip celice preračunala povprečno velikost in jih prikazala na grafu, za vsako število vozlišč posebej. V tem sklopu sva naredila še primerjavo povprečnih velikosti celic glede na število središč in vozlišč. Poleg tega pa sva se za mreže poiskala še gostoto porazdelitve, ki najbolj primerno opiše spreminjaje velikosti celice s spreminjajem velikosti mreže.

Rezultati

Binomska drevesa in cikli


Povprecne velikosti celic glede na število središč za binomska drevesa z 50
Povprecno število vozlišc v celici
40
30
20
10
0
10
20
30
Število središč
tip_celice
mala
povprecna
velika

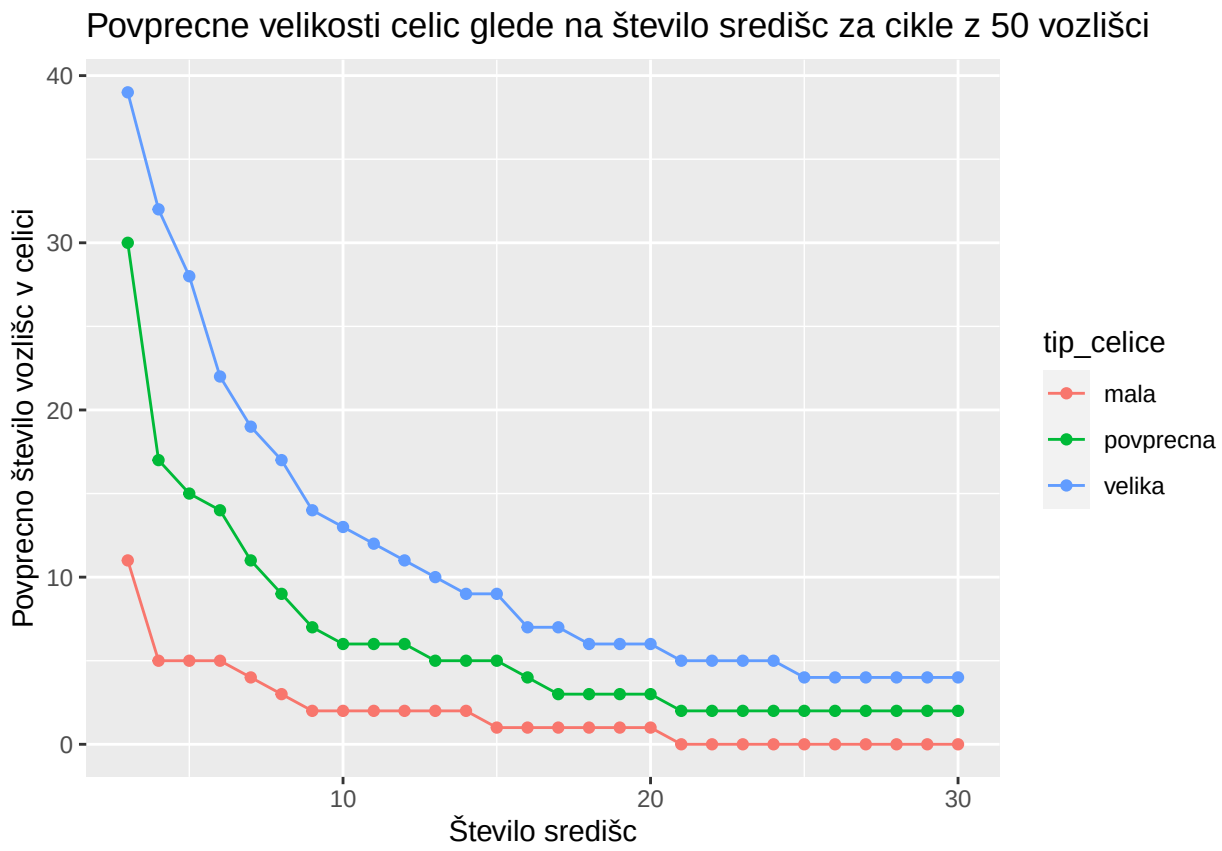

Povprecne velikosti celic glede na število središč za binomska drevesa z 10(
Povprecno število vozlišc v celici
0
20
40
20
40
60
Število središč
tip_celice
mala
povprecna
velika

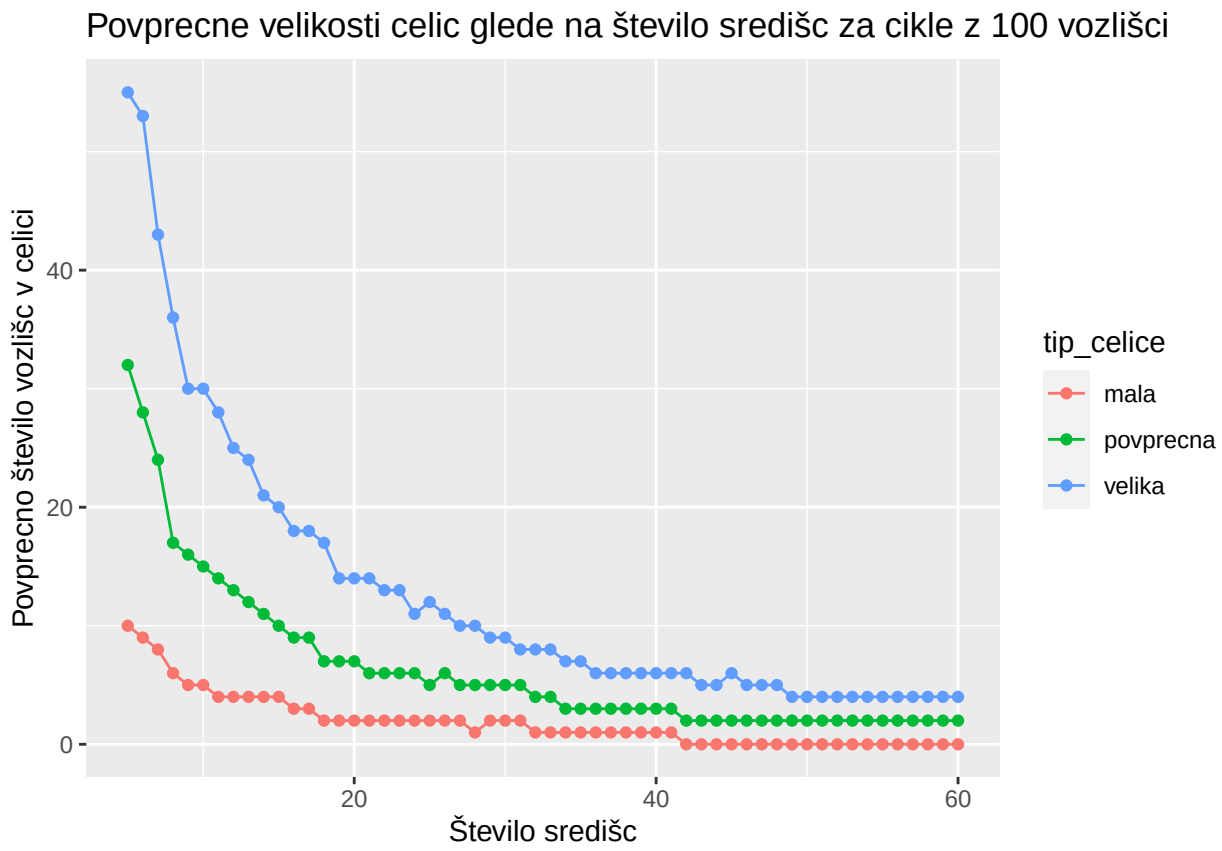


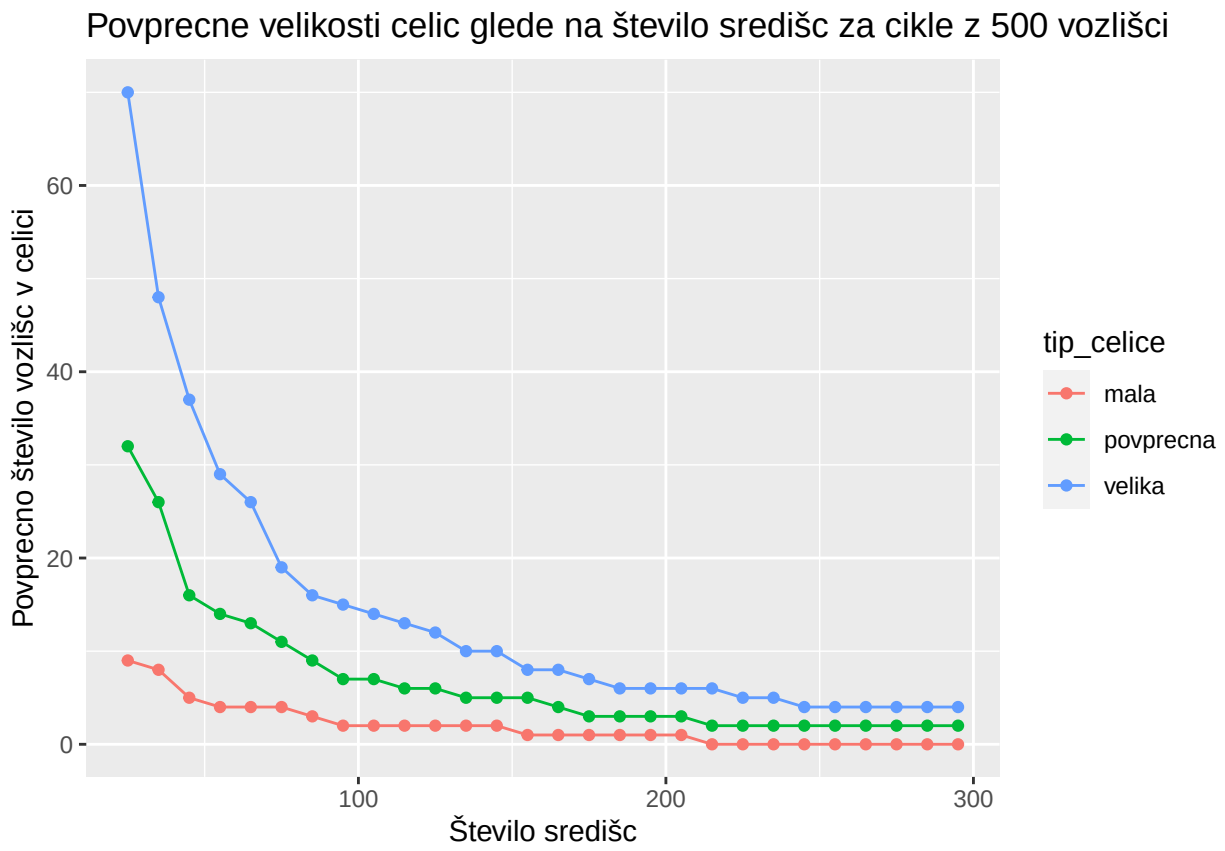
Kot je zelo lepo razvidno iz zgornjih grafov, se za vse 3 (in tudi vse neprikazane grafe) pojavlja isti vzorec. Ko je število središč majhno, so povprečne velikosti celice občutno večje kot ob velikih številih središč. To je seveda logično, saj je jasno, da je povprečno število celic sorazmerno padajoče z večanjem števila središč. Morda se na grafih nekoliko zavajajoče zdi, da imajo grafi s 100 in 500 vozlišči ob majhnem številu središč enako velikost celic, vendar je treba opomniti, da se graf za 500 vozlišč začne z 20 središči, graf za 100 vozlišči pa s 5, ko primerjamo pri enakem številu središč se vrednosti občutno razlikujejo - pri grafu s 100 vozlišči je pri dvajsetih središčih povprečna velikost povprečne celice nekaj manj kot 10, medtem ko je pri grafu z 500 vozlišči ta velikost 30. Na vseh grafih je očitno eksponentno padajoče sorazmerje, kar lahko predvidevamo za vse grafe z več vozlišči.

Zelo podobne rezultate dobimo tudi pri analiziranju ciklov. Diagrami imajo največjo povprečno velikost celic pri manjšem številu središč, ko pa to število povečujemo, pa velikosti celic eksponentno padajo. Opazimo lahko, da je tako kot pri prejšnjih grafih začetna razlika med velikimi, povprečnimi in majhnimi celicami velika, ko pa povečujemo število središč se razlike med njimi zmanjšujejo. Pri tem se pokaže predvsem razlika, da so najvišje povprečne velikosti celic pri ciklih za nekaj vozlišč večje (pri grafih z manj vozlišči, teh razlik skorajda ni, pri grafi z velikim številom vozlišm, recimo do 500, pa se razlikujejo za manj kot 10). Še vedno pa povprečne velikosti vseh tipov celic eksponentno padajo z višanjem števila središč.

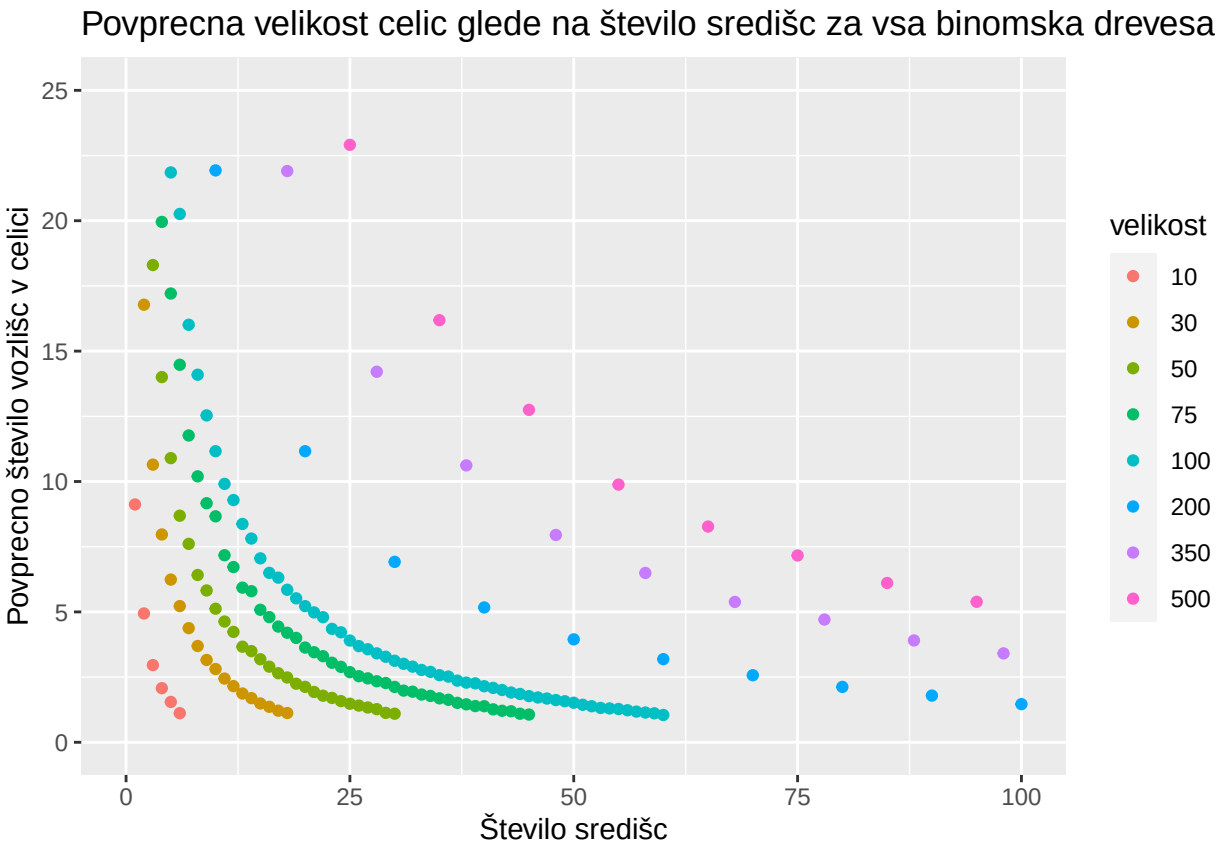
Iz tega lahko zaključimo, da dodajanje povezav v binomska drevesa za generiranje ciklov ne vpliva ključno pri majhnih številih vozlišč, verjetno pa bi se večji vplivi pokazali pri grafih z večjimi števili vozlišč (recimo od 1000 in več). Poleg tega lahko zaključimo, da se vsi tipi vozlišč obnašajo po podobnem principu - veliko število vozlišč za malo število središč in potem eksponentno padanje z večanjem števila središč.

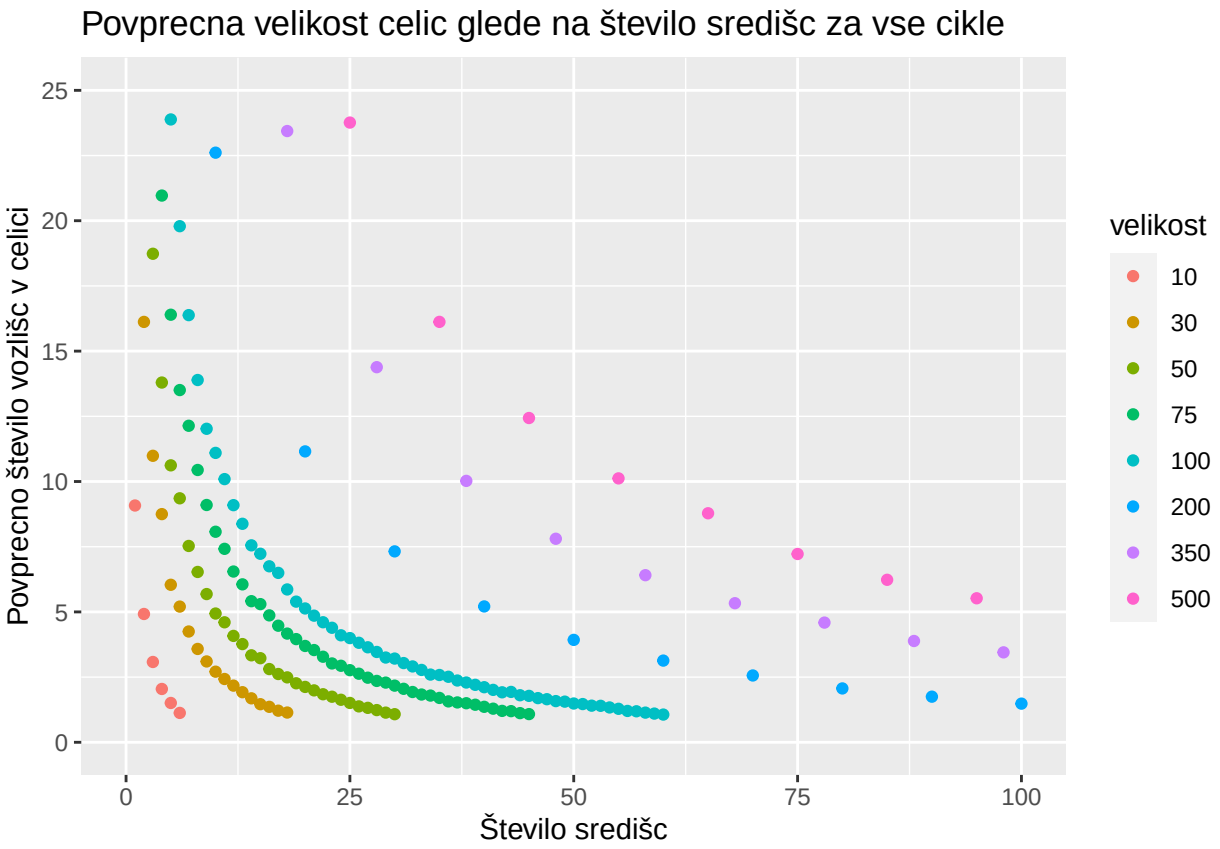

Povprecne velikosti celic glede na število središč za cikle z 50 vozlišci
Povprecno število vozlišc v celici
Število središč
0
10
20
30
40
10
20
30
tip_celice
mala
povprecna
velika


Povprecne velikosti celic glede na število središč za cikle z 100 vozlišči
Povprecno število vozlišc v celici
0
20
40
20
40
60
Število središč
tip_celice
mala
povprecna
velika

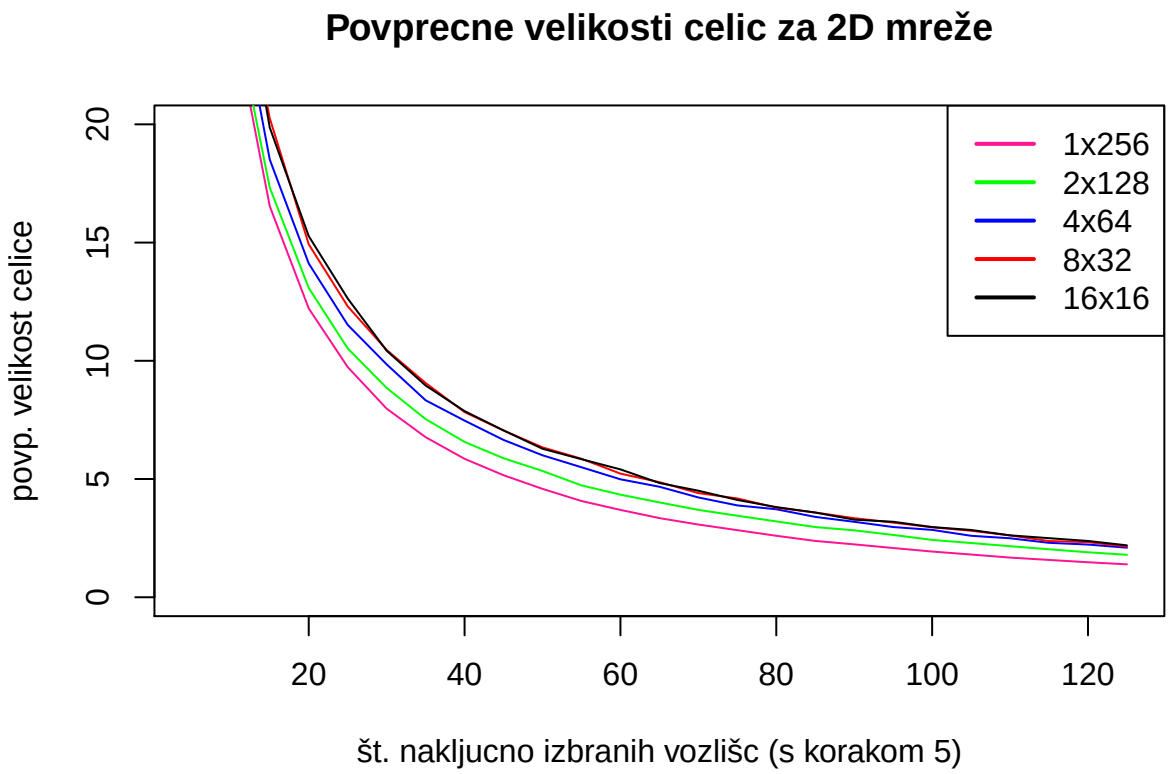


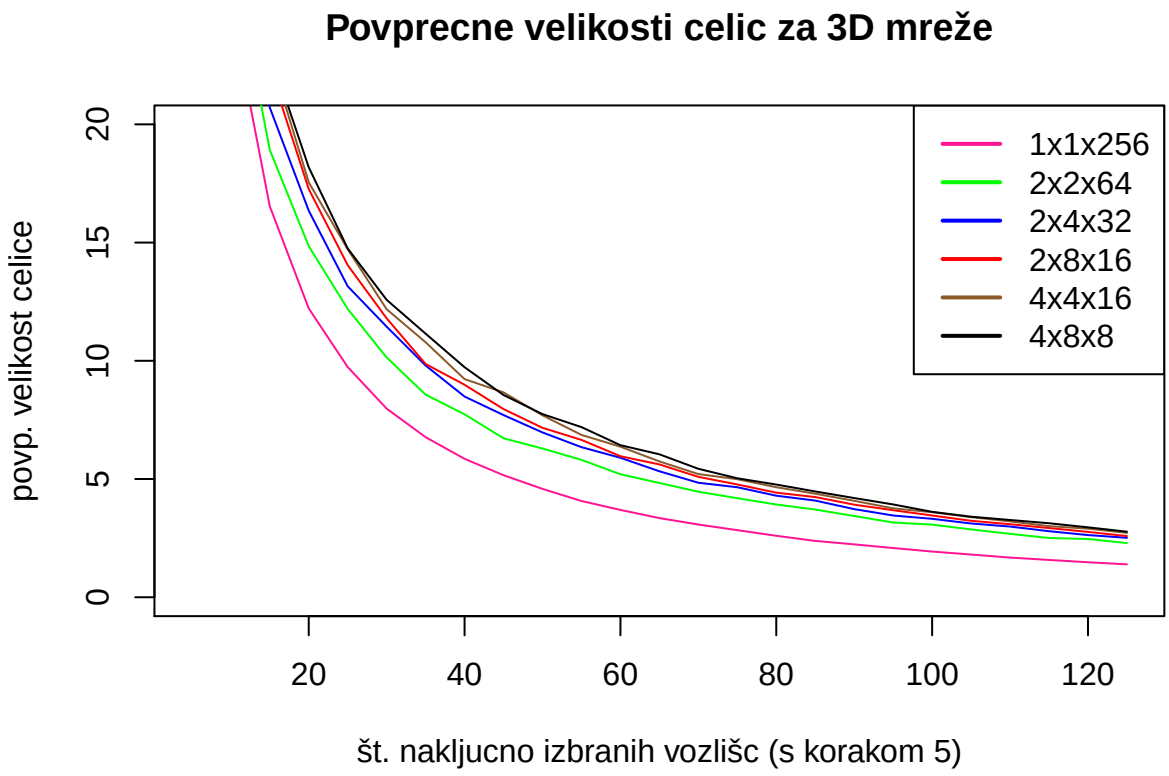
Ko primerjamo grafe za različna števila vozlišč, je eksponentno padanje povprečnega števila vozlišč lepo opazno - na začetku se povprečne velikosti celic zmanjšjejo hitro, na to pa so vedno bolj konstantne, potrebujejo večje število dodatnih središč za opazno zmanjšanje. Zelo dobro se vidi tudi razlika, da grafi z večjim števiom vozlišč, na primer z 500, dosežejo oziroma presežejo enako povprečno velikost celice kot grafi z manjšim številom vozlišč, pri precej večjem številu središč - graf s 500 vozlišči pri 25 središčih doseže podobno povprečno velikost celice (nekaj manj kot 25 vozlišč) kot graf s 100 vozlišči pri 5 središčih. Enaki trendi veljajo za vse opazovane tipe grafov, kot prikazujejo spodnje slike, razlikujejo se le v najvišji doseženi povprečni velikosti celice.


Povprecna velikost celic glede na število središč za vsa binomska drevesa
Povprecno število vozlišč v celici
Število središč
0
5
10
15
20
25
0
25
50
75
100
velikost
10
30
50
75
100
200
350
500



2D in 3D mreže Naslednja podatkovna struktura, ki sva si jo v luči analiziranja Voronoijevih diagramov pobliže ogledala, so dvo in tridimenzionalne mreže. Pri mrežah sva pričakovala, da se bodo povprečne velikosti Voronoijevih celic razlikovale od "strukture" mreže, saj lahko npr. mrežo s 24 vozlišči konstruiramo kot mrežo 3×8 ali kot 4×6. \ Za analizo sva tako izbrala vse možne mreže z 256 vozlišči (2^8). Vse možne mreže so torej, dvodimenzionalne: 1×256, 2×128, 4×64, 8×32 in 16×16; tridimenzionalne $2 \times 2 \times 64$, $2 \times 4 \times 32$, $2 \times 8 \times 16$, $4 \times 4 \times 16$ ter $4 \times 8 \times 8$. \ S korakom 5 sva dvajsetkrat generirala naključno izbrana vozlišča med 5 ter 125 vozlišči. Z grafa sva hitro ugotovila, da so dvodimenzionalne mreže precej podobne in da omejenost zgolj na pot dolžine 256 ne vpliva drastično na povprečne velikosti Voronoijevih celic. Tudi dodatna dimenzija na drugem grafu ne pomeni drastičnega povečanja povprečnih velikosti celic, npr. na mreži 1×256 nam 40 izbranih vozlišč generira Voronoijeve celice s povprečno močjo oz. velikostjo 5.85, medtem ko nam 40 izbranih vozlišč na najbolj "svobodni" mreži $4 \times 8 \times 8$ generira Voronoijeve celice povprečne moči 9.73.

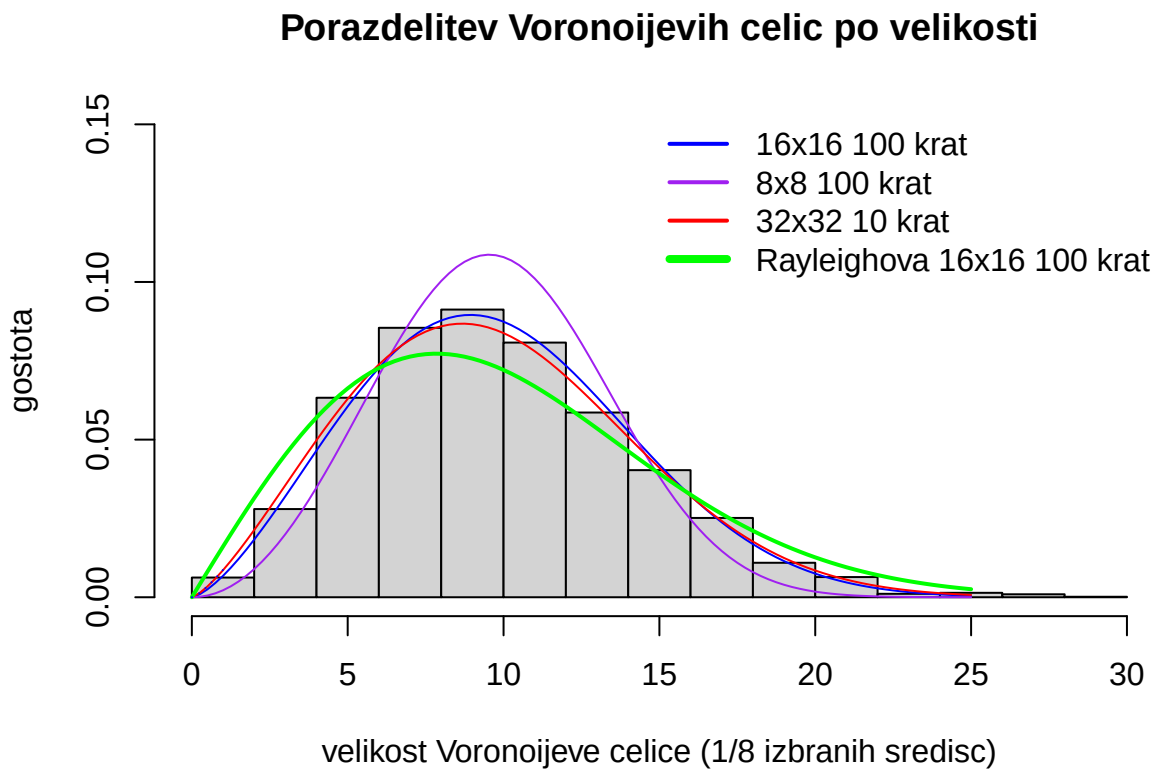

Povprecne velikosti celic za 2D mreže
povp. velikost celice
0
5
10
15
20
20
40
60
80
100
120
št. nakljucno izbranih vozlišc (s korakom 5)
1x256
2x128
4x64
8x32
16x16

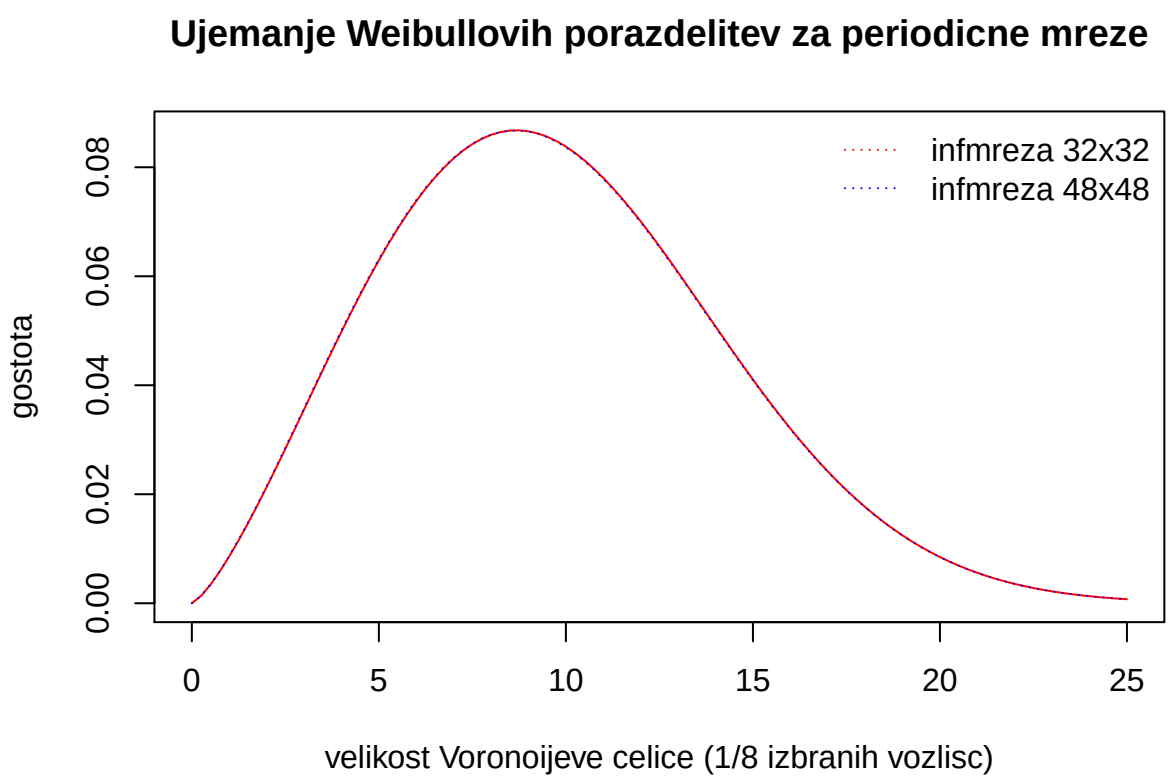


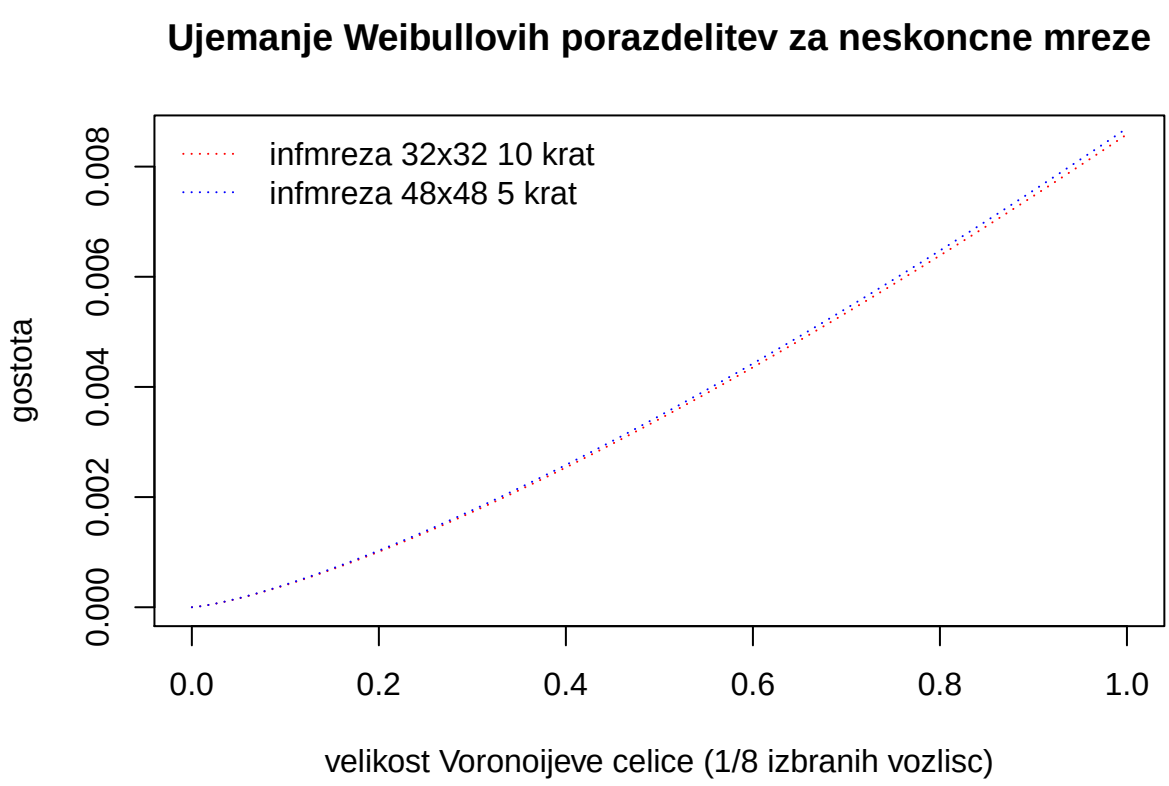
Simulacija neizbiranja robov Problem sva si poizkusila zastaviti še v drugačni luči. Kaj se zgodi, če ne dovolimo izbora robnih točk, torej takšnih, ki nimajo štirih povezav? V pythonu sva dodala novo funkcijo, s katero sva poizkusila simulirati neizbiranje robnih točk, to pa sva implementirala kar z dodajanjem povezav - dobljena funkcija nam tako generira grafe, podobne mrežam, le da ima vsaka vozlišče natanko 4 povezave. Intuitivno si to lahko predstavljamo tako, da smo vse robove mreže povezali med sabo, torej levi rob z desnim ter zgornjega s spodnjim.

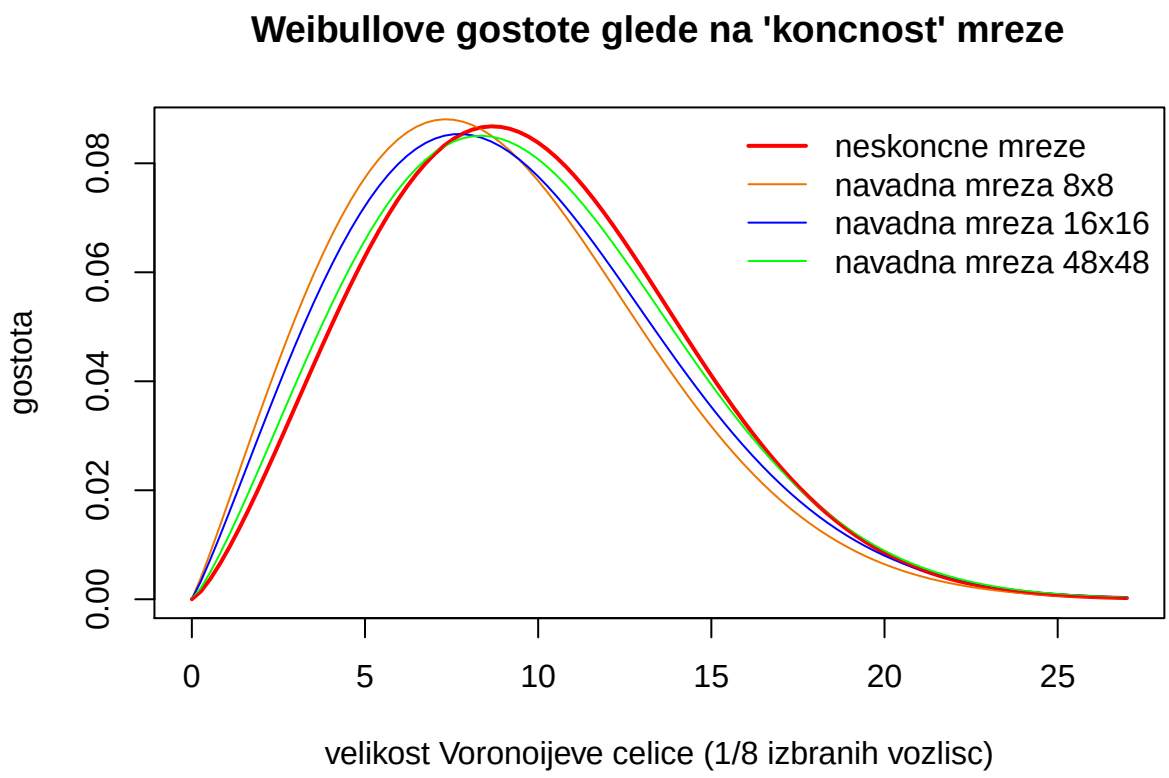
Na prvi sliki tako lahko vidimo histogram za 100 iteracij "neskončne" mreže velikosti 16×16 (t.j. take brez robov), kateremu sva po metodi najmanjših kvadratov dodala (modro) gostoto pripadajoče Weibullove porazdelitve. Da pokažemo dinamiko, sta na graf dodani tudi Weibullovi gostoti, dobljeni po stotih oz. desetih iteracijah na mreži 8×8 oz. 32×32. Opazimo lahko, da se gostoti dobljeni z večjih dveh mrež že precej lepo ujemata, gostota manjše, 8×8 mreže pa nekoliko odstopa. Mislimo si torej, da se s povečanjem št. vozlišč mreže Weibullove gostote še bolj ujemajo.
Na histogram je (z debelejšo zeleno) dorisana tudi Rayleighova krivulja za isti vzorec podatkov. Zakaj - Weibullova porazdelitev je odvisna od dveh parametrov, Rayleighova pa le od enega. Veljalo je razmisliti, da bi bil ta parameter lahko (fiksno) število izbranih Voronoijevih vozlišč (v obravnavanem primeru 1/8), a s slike vidimo, da se (modra) Weibullova gostota histogramu bolje prilega kot (zelena) Rayleighova.

Predpostavimo torej, da se za "neskončne" mreže porazdelitve za velike n ujemajo. Na sliki sta vidni pripadajoči Weibullovi gostoti za mreži 32×32 in 48×48. Ker se grafa skoraj ujemata, je dodana še tretja slika, kjer je skala močno pomanjšana, da lahko res opazimo ujemanje teh dveh gostot, obeh dobljenih na več kot 1000 podatkih.


Porazdelitev Voronoijevih celic po velikosti
16x16 100 krat
8x8 100 krat
32x32 10 krat
Rayleighova 16x16 100 krat
gostota
0.00
0.05
0.10
0.15
0
5
10
15
20
25
30
velikost Voronoijeve celice (1/8 izbranih sredisc)


Ujemanje Weibullovih porazdelitev za periodicne mreze
infmreza 32x32
infmreza 48x48
gostota
0.00
0.02
0.04
0.06
0.08
0
5
10
15
20
25
velikost Voronoijeve celice (1/8 izbranih vozlisc)


Ujemanje Weibullovih porazdelitev za neskoncne mreze
infmreza 32x32 10 krat
infmreza 48x48 5 krat
gostota
0.000
0.002
0.004
0.006
0.008
0.0
0.2
0.4
0.6
0.8
1.0
velikost Voronoijeve celice (1/8 izbranih vozlisc)



Vzemimo torej Weibullovo gostoto, dobljeno z metodo najmanjših kvadratov na desetih iteracijah "neskončne" mreže 32×32. To krivuljo po prejšnjem razmisleku proglasimo za gostoto vseh "mrež", torej takšnih brez robov. Na graf dodamo še krivulje navadnih, "končnih" mrež, torej takšnih z robovi. Vidimo, da se s povečanjem dimenzije navadne mreže Weibullove gostote vse bolj prilegajo rdeči krivulji, lahko bi rekli, da sva empirično dokazala, da je limitna funkcija zaporedja Weibullovih gostot navadnih mrež Weibullova gostota mreže brez robov.
Rezultat oz. potrditev hipoteze ni presenetljiva. Ko večamo dimenzijo mreže, se število notranjih vozlišč - takšnih s štirimi povezavami - povečuje s kvadratom, število robnih vozlišč - takšnih s tremi povezavami - pa pospešeno linearno. Za središča na robu so Voronoijeve celice seveda manjše, a ko v neskončnosti prevladajo notranja vozlišča, vpliv robov izgine.
Velja opozoriti na specifično izbiro tipov mrež oz. njihovih velikosti. Ker opazujeva dogajanja pri fiksnem izboru števila središč Voronoijevih celic, želiva, da je ta procent vozlišč celo število. Vzela sva torej takšne mreže, katerih osmina števila vozlišč je celo število, da sva minimizirala računske napake, ki bi nastale ob zaokroževanju.

Zaključek in možne izboljšave

Rezulati so pričakovani - povprečne velikosti celic padajo z večjim številom središč, za vse opazovane tipe grafov, skoraj gotovo pa tudi za katerekoli druge tipe. Bi pa se dalo verjetno precej stvari izboljšati. Zelo očitno je da bi na boljšo natančno rezultatov vplivalo večje število ponovitev (recimo 100 ali 1000) generiranja za vsak tip grafa, ter opazovanje grafov z večjim števiom vozlišč. To bi bilo v prvi vrsti verjetno možno s prekonstruiranjem in optimiziranjem kode programa, da bi ta delovala hitreje, saj je za velika števila vozlišč izjemno dolgotrajna. Na neki točki verjetno ne bi pomagalo niti to in bi enostavno potrebovali zmogljivejše računalnike za generiranje diagramov. Smiselno bi bilo opazovati tudi še katere druge tipe grafov, vendar se zaradi pomanjkanja časa za to nisva odločila, saj bi tako projekt zelo hitro postal zelo kompleksen.
Poleg tega bi bilo verjetno smiselno, da bi se nekoliko spremenilo generiranje ciklov. V trenutni kodi namreč

generiramo en naključni cikel iz binomskega drevesa za dano število vozlišč. Verjetno bi bilo za bolj natančne rezultate smiselno preveriti generiranje več različnih ciklov, da bi videli kako dodajanje povezav na različa mesta vpliva na povprečne velikosti celic. Tudi za to se nisva odločila, ker so tudi ostali tipi grafov zgenerirani enolično glede na število povezav, ter zaradi časovne stiske, vendar bi bilo to ob drugačnih pogojih vsekakor smiselno preveriti. Lahko bi dodala tudi grafične ponazoritve diagramov, da bi bile primerjave za velikosti Voronoijevih celic glede na različne tipe ter števila vozlišč in središč še bolj očitne in predstavljive.