UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Finančna matematika – 1. stopnja

Lara Vidmar in Žan Kastelic

Problem največjega pretoka

Finančni praktikum

Mentor: prof. dr. Sergio Cabello in asist. dr. Janoš Vidali

Ljubljana, 2021

Kazalo

1 Opis problema

Pri problemu največjega pretoka imamo pretočno omrežje, po katerem teče tekočina. Omrežje vsebuje dve oblikovani vozlišči, s in t. Iščemo največji možen pretok med njima. Vozlišče s se imenuje **izvor** in nima nobene vstopne povezave. Vozlišče t pa **ponor** in nima nobene izstopne povezave.

PODATKI:

- Imamo usmerjen graf $G = (V, E)$ z v naprej oblikovanima vozliščema s in t. Pri tem je V množica vozlišč in E množica povezav v grafu.
- Na vsaki povezavi $(v_i, v_j) \in E$ imamo nenegativno realno število c_{ij} tj. prepustnost ali kapaciteta povezave (v_i, v_j). Prepustnost c_{ij} lahko razširimo na vse pare vozlišč: $c(i,j) = \begin{cases} c_{ij}, & \text{če } (v_i, v_j) \in \text{E} \\ 0, & \text{če } (v_i, v_j) \notin \text{E} \end{cases}$
- Urejeno četvorko (G, s, t, c) imenujemo **pretočno omrežje.**

Pri zgornjih podatkih iščemo največji pretok, ki je preslikava $f : V \times V \to \mathbb{R}$. $f(i,j) < 0$, pomeni tok $|f(i,j)|$ od j proti i. Pri tem mora bit zadoščeno naslednjim pogojem.

- *Ustreznost pretoka*: $f(i,j) \leq c(i,j)$ za $\forall i, j \in V$
- *Antisimetričnost pretoka*: $f(i,j) = -f(i,j)$ za vse $i, j \in V$
- *Kirchhoffovi zakoni*: $\sum i \in V f(i,j) = 0$ za vse $j \in V \setminus \{s,t\}$

Velja $f(i,j) = -f(i,j) = 0$. Če je f pretok, je povezava $(v_i, v_j) \in E$ **zasičena,** če velja $f(i,j) = c(i,j)$. Oziramo **nenasičena**, če je $f(i,j) < c(i,j)$.

Velikost pretoka f je

$$|f| = \sum_{i \in V} f(i,t).$$

2 Načrt dela

Najinega projekta se bova lotila v programu R. Najprej bova implementirala algoritem za iskanje največjega pretoka v acikličnem povezanem grafu. Pomagala si bova z Edmonds-Karp algoritmom. Algoritem najprej poišče poti od izvora s do ponora t, z uporabo iskanja v širino, potem pa s Ford-Falkersonovega algoritmom poišče največji pretok. Midva bova napisala algoritem, ki bo prvo izpisal vse poti, potem pa bova gledala minimum prepustnosti določene poti. Kot sva ugotovila obstaja v programu R že vgrajena funkcija za iskanje največjega pretoka tj. maxFlowFordFulkerson. Poiskušala bova primerjat algoritma in ugotoviti kateri je bolj učinkovit.

Grafe bova implementirala s pomočjo matrike sosednosti, kjer bodo prepustnosti izbrane naključno iz že v naprej podanega intervala celih števil. Če utež ne bo enaka 0, potem bo ta povezava obstajala. Za lažjo predstavo bo funkcija graf tudi narisala.

Kot drugo bova poizkušala graf implementirati s pomočjo naključnih geometrijskih grafov. Neka povezava bo obstajala, če bo njena povezava manjša

ali enaka od nekega števila r. Tem povezavam bova določila smer in nato še naključne uteži.

Eksperimente bova delala na zgoraj definiranih grafih, ki jim bova odstranila eno ali več vozlišč, odstranila povezave, zamenjala smer povezave. Opazovala bova kako se pretok spreminja in poizkušala iz tega dobiti kakšno lastnost oz. bova opisala opažanja.

3 Opis programa

Najprej sva generirala matriko, v kateri so generirani podatki grafa. Pri tem so elementi matrike uteži, ki so naključne. Kombinacija vrstice in stolpca pa predstavlja povezavo (na primer (vrstica 4, stolpec 6) pomeni povezava iz oglišča 4 v 6). Ker pri iskanju pretoka potrebujeva acikličen graf, so v matriki v prvem stolpcu, v zadnji vrstici in na diagonali same ničle. Iz nje sva potem generirala graf. Da bi lahko napisala funkcijo, ki bo iskala največji pretok, sva potrebovala tabelo, v kateri bodo v prvem stolpcu vstopna vozlišča povezave, v drugem izstopna vozlišča povezave, v tretjem pa uteži. To sva naredila v funkciji oceti_in_sinovi. Najvecji pretok sva poiskala tako, da sva locirala pot od izvora (vozlišče 1) do ponora (vozlišče z največjo številko) in skoz njo spustila največi možen pretok. Potem sva matriko posodobila, tako da sva ostranila povezave z ničelno utežjo in postopek ponovila. Kot sva ugotovila, je ta algoritem delal zelo počasi in je izračunal pretok za grafe do 7 vozlišč.

Zato sva generirala nov algoritem za iskanje pretoka, pri tem sva si pomagala psevdokodo Edmonds-Karp algoritma, kot je prikazana na sliki 1.

```
algorithm EdmondsKarp is
    input:
        graph   (graph[v] should be the list of edges coming out of vertex v in the
                 original graph and their corresponding constructed reverse edges
                 which are used for push-back flow.
                 Each edge should have a capacity, flow, source and sink as parameters,
                 as well as a pointer to the reverse edge.)
        s       (Source vertex)
        t       (Sink vertex)
    output:
        flow    (Value of maximum flow)

    flow := 0   (Initialize flow to zero)
    repeat
        (Run a breadth-first search (bfs) to find the shortest s-t path.
         We use 'pred' to store the edge taken to get to each vertex,
         so we can recover the path afterwards)
        q := queue()
        q.push(s)
        pred := array(graph.length)
        while not empty(q)
            cur := q.pull()
            for Edge e in graph[cur] do
                if pred[e.t] = null and e.t ≠ s and e.cap > e.flow then
                    pred[e.t] := e
                    q.push(e.t)

        if not (pred[t] = null) then
            (We found an augmenting path.
             See how much flow we can send)
            df := ∞
            for (e := pred[t]; e ≠ null; e := pred[e.s]) do
                df := min(df, e.cap - e.flow)
            (And update edges by that amount)
            for (e := pred[t]; e ≠ null; e := pred[e.s]) do
                e.flow  := e.flow + df
                e.rev.flow := e.rev.flow - df
            flow := flow + df

    until pred[t] = null  (i.e., until no augmenting path was found)
    return flow
```

Slika 1: Psevdokoda Edmonds-Karp algoritma.

Najprej sva spremenila najino matriko v igraph, kateri je veliko bolj učinkovit. Napisala sva algoritem pregled_v_sirino, ki išče poti od izvora do ponora. S pomočjo tega pa sva napisala edmonds_karp algoritem. Pri tem sva za odstranjevanje povezav in lociranje minimalnih utezi uporabljala vgrajene funkcije od igrapha.

Kot sva že napovedala v načrtu dela, sva generirala geometrijske grafe. Razdelila sva jih na tri tipe in sicer, geometrijski grafi, ki imajo za uteži kar razdalje med točkami, potem grafe, ki imajo za uteži inverz razdalj in še zadnje, ki imajo naključne uteži. Pri tem sva morala bit previdna, da je točka, ki je najbližje izhodišču markirana z 1, tista najdlje oddaljena pa je označena s številom točk. Velikost grafa je odvisna od števila r, ki predstavlja razdaljo med točkami. Večji kot je r več povezav bo v grafu.

4 Generiranje podatkov

Podatke sva najprej generirala za navadne grafe, ki sva jih pridobila iz matrike, potem pa še za geometrijske grafe. Pogledala sva kako se pretok spreminja, če odstranimo povezavo, ki ima minimalno utež in kako, če odstranimo povezavo z maksimalno utežjo. Potem pa še, kako se spremeninja pretok, če odstraniva naključno točko, ki ni izvor oziroma ponor. Funkcije sva generirala tako, da nama vrne tabelo, ki prikazuje pretok, kjer so vrstice vrednosti za različno število točk, v stolpcih pa koliko povezav oz. koliko točk sva odstranila. Pri navadnih grafih sva gledala, kaj se dogaja s pretokom, če so vse uteži enake. Le pri tem sva ugotovila nek algoritem, katerega sva tudi napisala.

Pri geometrijskih grafih sva funkcije generirala na podoben način. Pri funkcijah, ki odstranjujejo povezave in točke, sva vse vrste geometrijskih grafov združila v eno funkcijo. In sicer tako, da sva v argument dodala tudi *tip*, kateri predstavlja vrsto geometrijskega grafa, tj. igraf_razdalje_so_utezi pri tem napišemo $tip = 1$, igraf_razdalje_so_inverz je $tip = 2$ in še igraf_utezi_so_nakljucne, ki je $tip = 3$.

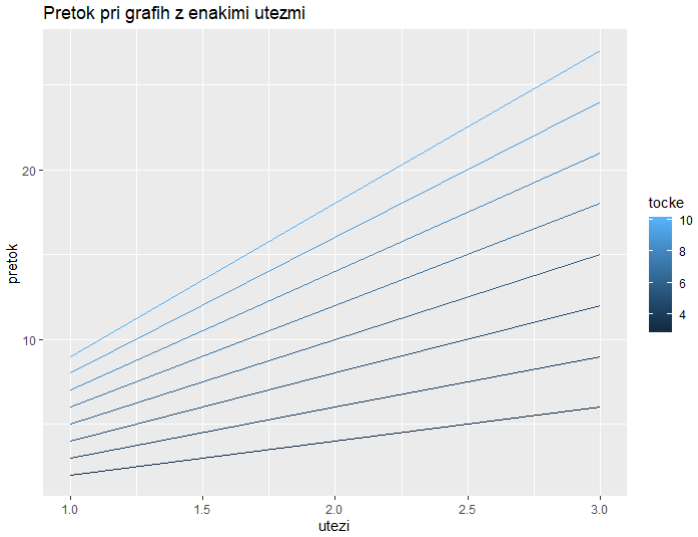
Ker so geometrijski grafi zelo odvisni od razdalje r, naju je zanimalo tudi kako se spreminja pretok, če spreminjava r.

5 Opis in razlaga eksperimentov

Eksperimente sva delala na grafih z 10 točkami in gledala njihov pretok. Podatke bova predstavila v grafih, saj so tako najbolj razvidne spremembe. Ker večina grafov generirava naključno, je pretok veliko odvisen tudi od naklučja uteži in povezav.

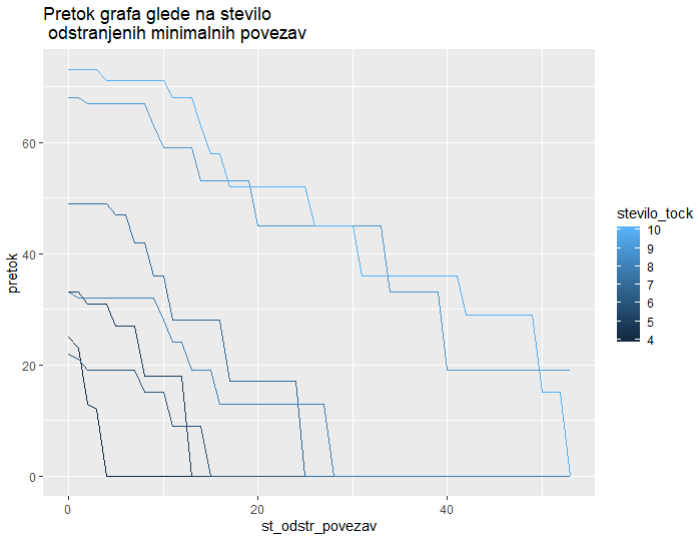
5.1 Grafi generirani s pomočjo matrike.

• Na grafu 2 je razvidno, da pri grafih, kjer imajo vse povezave iste uteži, se pretok povečuje linearno z večanjem uteži in večanjem števila točk.



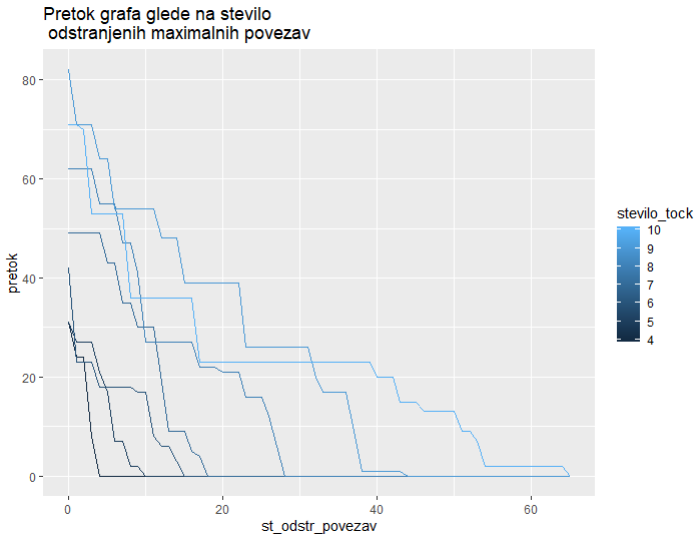
Slika 2: Pretok na grafih z enakimi utežmi.

• Na grafu 3 je prikazano spreminjanje pretoka, če grafu odstranjujemo povezavo z minimalno utežjo. Kot pričakovano je pretok vsakič manjši.



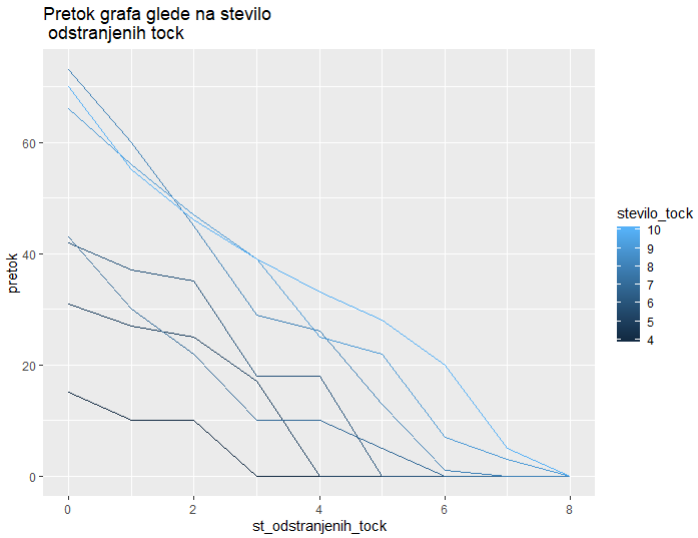
Slika 3: Pretok na grafih z naključnimi utežmi, odstranimo povezavo z min. utežjo.

• Na grafu 4 je prikazano kako se pretok spreminja, če grafu odstranjujemo povezavo z največjo utežjo. Graf je podoben zgornjemu, le da se pri tem pretok veliko hitreje manjša v primerjavi z zgornjim grafom.



Slika 4: Pretok na grafih z naključnimi utežmi, odstranimo povezavo z max. utežjo.

• Na grafu 5 je prikazano spreminjanje pretoka, če odstranjujemo točke. Vsak korak odtstranimo eno točko več. Ta graf je veliko bolj linearen od zgornjih dveh, saj z odstranitvijo ene točke lahko odstranimo veliko več povezav kot samo eno povezavo.



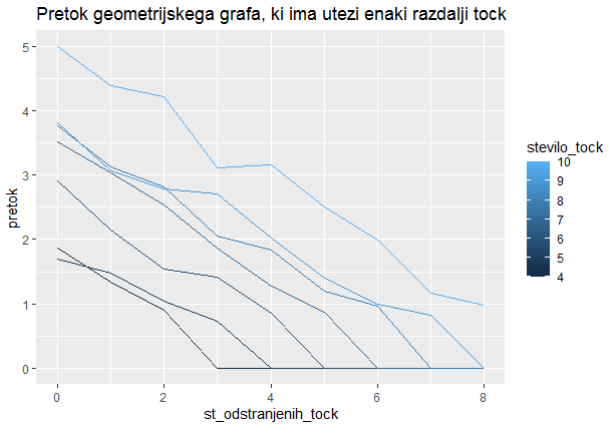
Slika 5: Pretok na grafih z naključnimi utežmi, odstranimo naključne točke.

5.2 Geometrijski grafi

Sledijo podatki za geometrijske grafe. Ločeni so glede na tipe uteži.

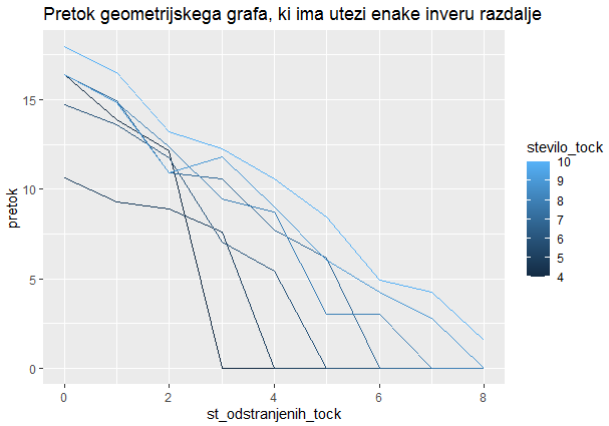
→ Geometrijski grafi, ki jim odstranjujemo točke.

∗ Na grafu 6 je prikazan geometrijski graf, ki ima za uteži kar razdaljo med točkami. Kot vidimo pretok linearno pada. Vsakič, ko odstranimo eno točko več je pretok seveda manjši.



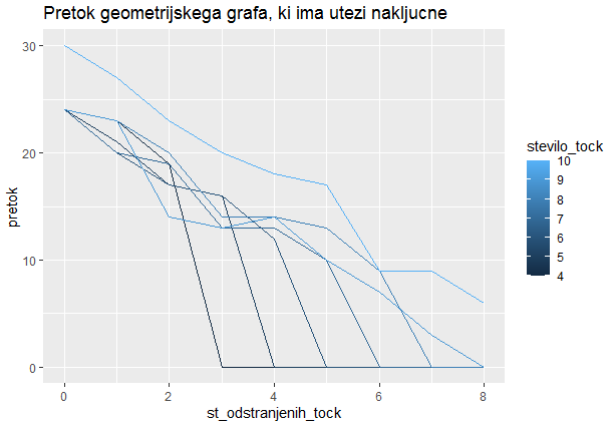
Slika 6: Pretok pri geometrijskem grafu ($tip = 1$), ki mu odstranjujemo točke.

∗ Na grafu 7 je prikazan geometrijski graf, ki ima uteži inverz razdalj med točkami. Kar pomeni točki, ki sta bližje skupaj bosta imeli večjo utež, kot pa tisti, ki sta bolj oddaljeni. Če graf pogledamo brez nenadnih skokov v pretoku, kateri nastanek bova objasnila kasneje, je nekako podoben prejšnjemu. Le da je pri tem pretok veliko večji že od začetka.



Slika 7: Pretok pri geometrijskem grafu ($tip = 2$), ki mu odstranjujemo točke.

∗ Na grafu 8 je prikazan geometrijski graf, ki ima za uteži naključna cela števila do 10. Pri tem, poleg veliko večjega začetnega pretoka, ki je posledica večjih uteži, ne opazimo razlike od zgornjih dveh. Pretok se z vsako odstranjeno točko zmanjša.

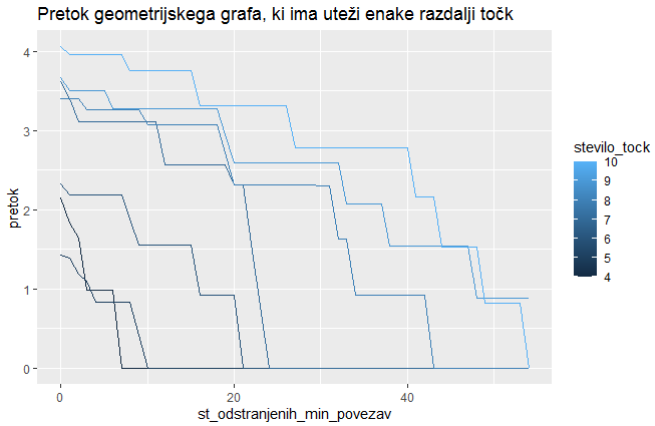


Slika 8: Pretok pri geometrijskem grafu ($tip = 3$), ki mu odstranjujemo točke.

Lahko bi rekli, da je ne glede na to, kakšen geometrijski graf vzamemo, pretok padajoč glede na število odstranjenih točk.

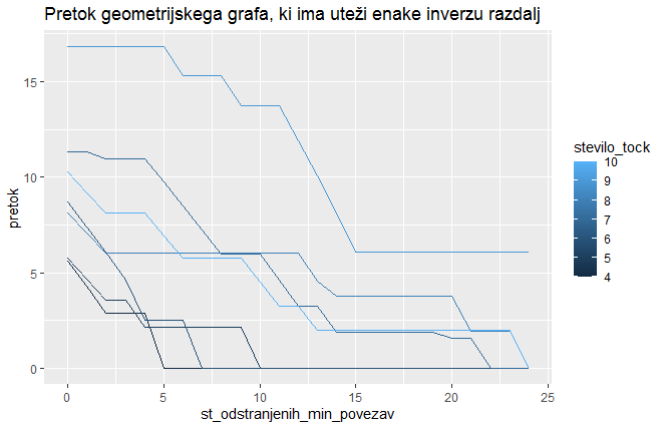
→ Geometrijski grafi, ki jim odstranimo minimalno utež na povezavi.

∗ Na grafu 9 je prikazan geometrijski graf, ki ima za uteži kar razdaljo med točkami. Odstranjujemo mu minimalno povezavo. Pretok je padajoč. Če povezava, ki je bila odstranjena, ni bila na poti od izvora do ponora, bo seveda pretok ostal nespremenjen, zato je pri tem velikokrat ravna črta na določeni višini. Drugače pa se slej ko prej pretok približa ničli.



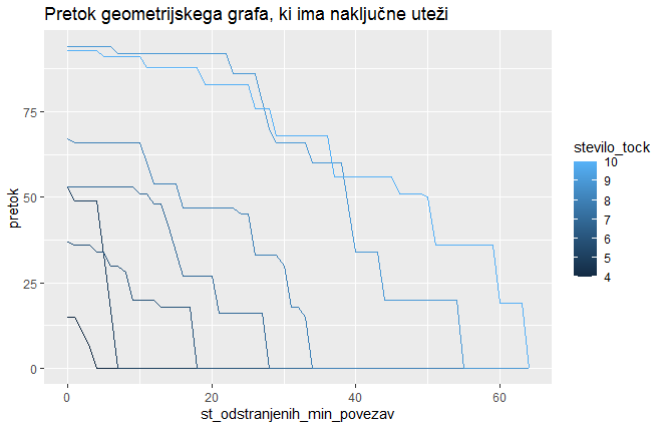
Slika 9: Pretok pri geometrijskem grafu ($tip = 1$), ki mu odstranjujemo minimalno utež na povezavi.

∗ Na grafu 10 je prikazan geometrijski graf, ki ima za uteži inverz razdalj med točkami. Pri tem je zgodba podobna kot pri prejšnem, le da je tukaj pretok veliko večji, zaradi večjih uteži.



Slika 10: Pretok pri geometrijskem grafu ($tip = 2$), ki mu odstranjujemo minimalno utež na povezavi.

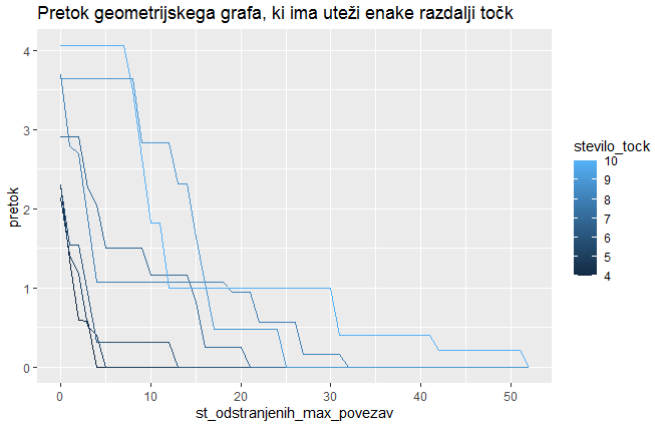
∗ Na grafu 11 je prikazan geometrijski graf, ki ima za uteži naključna cela števila do 10. Tukaj je padanje veliko bolj konsistentno. To bi lahko pripisali bolj enakomerno razporejenim utežem v grafu, saj ko generiramo graf z celimi števili do 10, je možnosti za različne uteži bolj malo.



Slika 11: Pretok pri geometrijskem grafu ($tip = 3$), ki mu odstranjujemo minimalno utež na povezavi.

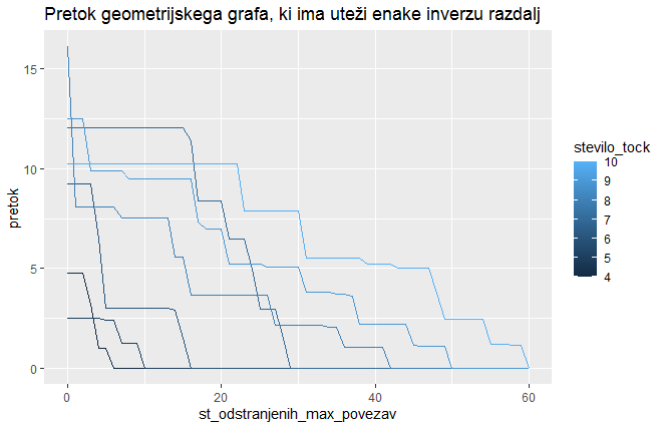
→ Geometrijski grafi, ki jim odstranjujemo maksimalno utež na povezavi.

∗ Na grafu 12 je prikazan geometrijski graf, ki ima za uteži kar razdaljo med točkami. Ker odstranjujemo maksimalno utež, pretok veliko hitreje pride do 0.



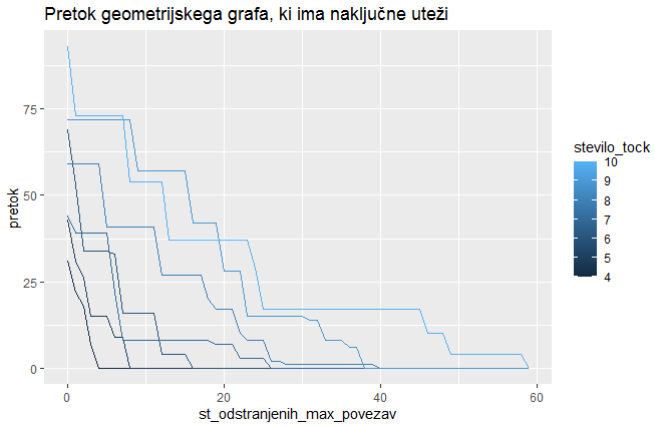
Slika 12: Pretok pri geometrijskem grafu ($tip = 1$), ki mu odstranjujemo maksimalno utež na povezavi.

∗Na grafu 13 je prikazan geometrijski graf, ki ima uteži inverz razdalj med točkami. Enako situacijo imamo tudi tukaj, le da začnemo z večjim pretokom. Je pa padanje nekoliko počasnejše.



Slika 13: Pretok pri geometrijskem grafu ($tip = 2$), ki mu odstranjujemo maksimalno utež na povezavi.

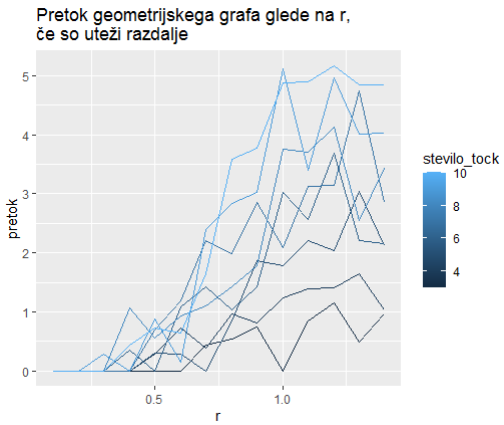
∗ Na grafu 14 je prikazan geometrijski graf, ki ima za uteži naključna cela števila do 10. Pretok je glede na pretekla grafa večji, drugače pa podoben kot zgoraj.



Slika 14: Pretok pri geometrijskem grafu ($tip = 3$), ki mu odstranjujemo maksimalno utež na povezavi.

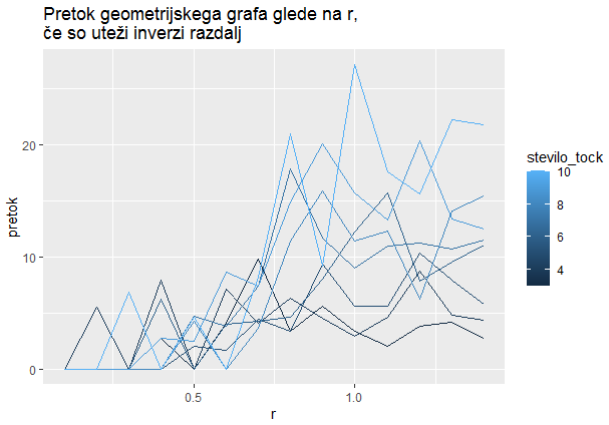
→ Geometrijski grafi, glede na spreminjanje razdalje r. V grafu so tiste povezave katera razdalja je manjša od r. Gledamo kako se spreminja pretok, ko r teče od 0.1 do $\sqrt{2}$, z razmikom 0.1.

∗ Na grafu 16 je prikazan geometrijski graf, ki ima za uteži kar razdaljo med točkami. Kot je razvidno je pri majhnem r, pretok v večini primerih enak 0, saj je takrat zelo malo povezav v grafu. Ko se r povečuje, pa je seveda pretok večji. Največji pretok naj bi imel graf z desetimi točkami, najmanši pa s štirimi. Kot vidimo tukaj ni ravno tako, v grafu imamo veliko skokov, katero posledico bova opisala kasneje.



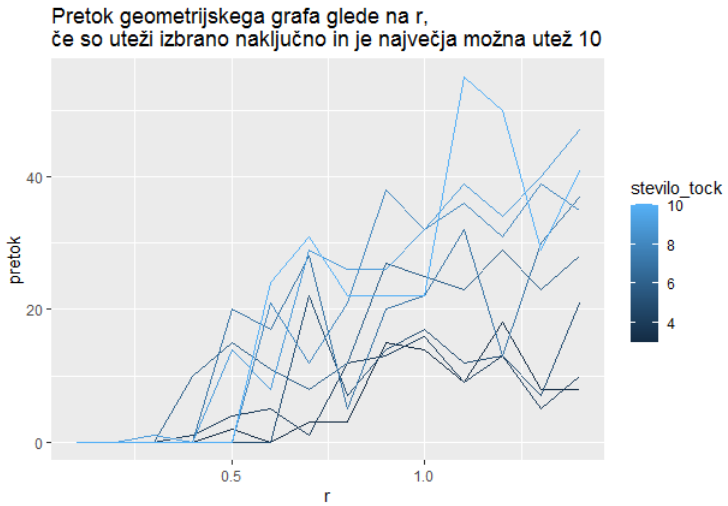
Slika 15: Pretok pri geometrijskem grafu ($tip = 1$), kjer spreminjamo r.

∗ Na grafu 16 je prikazan geometrijski graf, ki ima uteži inverz razdalj med točkami. Tukaj so skoki še bolj očitni, saj ima npr. graf s 4 točkami pri $r = 0.3$ večji pretok, kot graf pri $r = 0.4$. Vendar kljub temu lahko sklepamo, da se s povečevanjem r pretok veča.



Slika 16: Pretok pri geometrijskem grafu ($tip = 2$), kjer spreminjamo r.

∗ Na grafu 17 je prikazan geometrijski graf, ki ima za uteži naključna cela števila do 10. Kot zgoraj je graf malo napačen, vendar je sklep isti. Grafi z manj točkami imajo manjši pretok, kot grafi z več. Graf z istim številom točk ima pri manjšem r manjši pretok, kot pri velikem r.



Slika 17: Pretok pri geometrijskem grafu ($tip = 3$), kjer spreminjamo r.

Če povzamemo vse skupaj, lahko vidimo, da se pretok zmanjšuje, ne glede na to, ali odstranimo točke ali povezave. Razlika je le v tem, kako hitro pride do ničelnega pretoka. Seveda je to odvisno tudi od posameznega grafa. Pri spreminjanju razdalje med točkami, pa je pri večjem r pretok večji.

Naj omeniva še pomankljivost, ki sva jo opazila in je predvsem v grafih, kjer spreminjava r zelo očitna. Vemo, da bi moral biti pretok pri grafu z 10 točkami, v večini primerih večji, kot pri grafu z manj točkami. Enako, če grafu odstranimo povezavo, bo pretok enak oziroma manjši, kar pa pri nama ne drži nujno. Opazimo lahko, da imajo pri najinih grafih ponekod grafi z manj točkami večji pretok kot grafi z več točkami. Še najbolj očitno je to pri spreminjanju razdalje r. Graf z večjim r ima manjši pretok, kot graf z majhnim r, kar pa ni logično. To je zato ker, sva narobe generirala podatke. Se pravi, ko sva odstranila točko oziroma povezavo, sva graf ponovno zgenerirala, seveda z manj točkami in povezavami, vendar zopet naključno. To je pripeljalo do takih skokov v grafu. Bolj relavantno bi bilo gledati pretok istega grafa in temu odstranjevat točke oziroma povezave. Vendar kljub tej napaki lahko iz grafov vidimo, kako se spreminja pretok, ob spreminjanju različnih parametrov grafa.

5.3 Hitrost funkcij

Kot sva napisala zgoraj, naju je zanimala tudi učinkovitost najine funkcije. Že takoj sva ugotovila, da je najina prva funkcija za pretok neučinkovita. Medtem ko je edmond_karp v primerjavi z vgrajeno funkcijo maxFlowFordFulkerson tudi bolj neučinkovit, kar je bilo tudi za pričakovati. V spodnji preglednici imava naveden čas v sekundah, kako hitro je izračunal pretok pri različnem številu točk v grafu. Za primerjavo: vgrajena funkcija za izračun pretoka na 30 točkah potrebuje 0.11 sekunde.

st_tock	*edmonds_karp*	*maxFlowFordFulkerson*
3	0.16	0
4	0.19	0
5	0.17	0
6	0.20	0
7	0.25	0
8	0.36	0
9	0.29	0
10	0.33	0
11	0.45	0