UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Finančna matematika – 1. stopnja

Martin Praček in Mela Malej

Graffiti Conjecture 194

Seminarska naloga

Mentor: Riste Škrekovski

Ljubljana, 2019

Kazalo

Graffiti Conjecture 194

Povzetek

Za predmet Finančni praktikum v študijskem letu 2018/19 Martin Praček in Mela Malej dobila nalogo predstaviti problem Grafittijeve konjekuture 194.
To nalogo sva naredila s splošno kodo ter genetskim algoritmom.

1. Predstavitev problema

Grafittijeva konjektura 194 nam zastavi vprašanje iz teorije grafov.Zanima nas, ali za vsak preprost, povezan graf, ki zadošča pogoju

$$\alpha(G) \leq 1 + \lambda_{avg}(G),$$

velja, da obstaja hamiltonska pot. Gre za računalniško generirano trditev, pri kateri nas zanima, ali lahko najdemo protiprimer, kar je tudi najina glavna naloga.

2. Pogoji našega problema

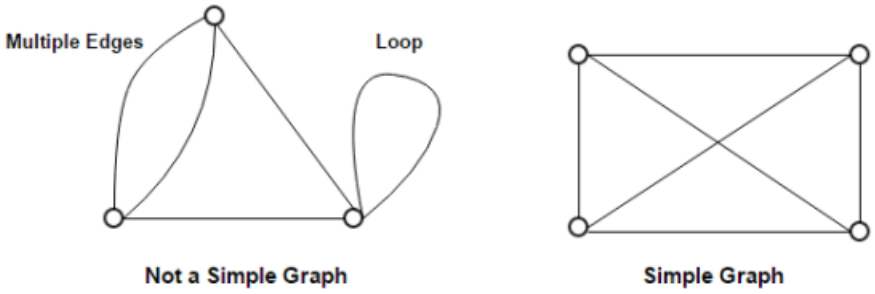
Naš pogoj je torej

$$\alpha(G) \leq 1 + \lambda_{avg}(G),$$

kjer je G naš graf, $\alpha(G)$ je neodvisnostno število, λ_{avg} pa je povprečna lokalna neodvisnost grafa.

Neodvistnostno število grafa $\alpha(G)$ nam pove moč največje množice, ki vsebuje vozlišča grafa G, od katerih nobeni dve niste sosednji.

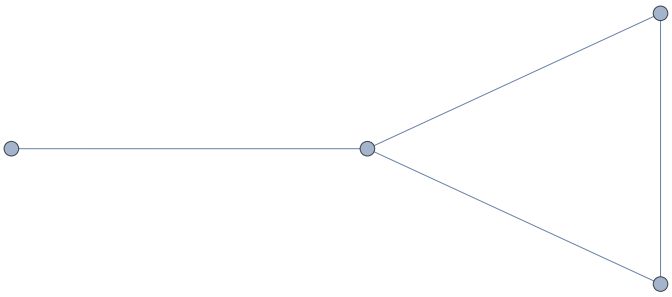
Lokalna neodvistnost grafa $\lambda(G, v)$ nam pove neodvistnostno število podgrafa Gv grafa G, kjer je Gv definiran na sosedih vozlišča v. $\lambda_{avg}(G)$ nam pove povprečno lokalno neodvistnost.



Ves čas zahtevamo, da je graf **preprost in povezan**. Ta zahteva pomeni, da ne obstaja vozlišče, ki ne bi bilo sosednje vsaj enemu drugemu vozlišču, med dvema povezavama obstaja kvečjemu ena povezava, poleg tega pa ne obstaja povezava od vozlišča v do vozlišča v, kar pomeni da ne obstajajo zanke na enem vozlišču.

Graf ima **hamiltonsko pot**, če obstajata dve vozlišči, ki ju povezuje pot, ki natančno enkrat obišče vsako vozlišče grafa.

Najina naloga je torej na dovolj velikem vzorcu grafov pokazati da to velja, ali pa dobiti protiprimer, ki bo dokazal da to ne drži.



2.1. **Primer.** Za lažje razumevanje je tukaj še poseben primer, ki mu moramo določiti neodvisnostno število ter povprečno lokalno neodvisnost.

Očitno je neodvistnostno število 2, povprečna lokalna neodvistnost pa je

$$\frac{1+3+1+1}{4} = \frac{6}{4}.$$

Naša neenakost torej pravi, da je

$$2 \leq 1 + \frac{6}{4},$$

kar drži. Naša naloga sedaj je dokazati, da hamiltonska pot ne obstaja. Vendar nam to v našem primeru ne bo uspelo, saj jo lahko precej enostavno najdemo.

3. Delo

Z najinim delom sva pričela v programskem jeziku Python, vendar sva hitro ugotovila sizifovstvo tega početja, in vse prestavila v programski jezik Sage, kjer sva v CoCalc spisala najino kodo v Jupyter Notebook.

4. Koda in ideja programa

Glavna ideja najine kode sledi glavni ideji najine naloge. Po vrsti sva razvrstila vse koščke neenakosti, in nato pogledala, če za primerne grafe le ta tudi velja. Tako sva dobila vrsto pomožnih ter eno glavno funkcijo.

4.1. **Pomožne funkcije.** Kot že rečeno, sva v okvirju najinega dela uporabila vrsto pomožnih funkcij.

4.1.1. *neodvistnostno_stevilo(G).* Ta funkcija nam vrne neodvistnostno število grafa G. Tukaj smo si pomagali z funkcijo *.independent_set*(), ki vrne največjo množico, kjer nobeni vozlišči nista sosednji.s

```
def neodvisnostno_stevilo(G):
    """
    Vrne neodvistno število grafa G
    Za pomoc uporabimo independet_set() iz modula neusmerjenih grafov.
    """
    neodvisno = G.independent_set()
    dolzina = len(neodvisno)
    return dolzina
```

4.1.2. *lokalna_neodvistnost(G, vozlisce).* Ta funkcija nam vrne lokalno neodvistnost grafa G.

```
def lokalna_neodvisnost(G, vozlisce):
    """
    Vrne lokalno neodvistnost grafa G v vozliscu v.
    Lokalna neodvistnost je neodvistnost podgrafa, ki ga določajo sosedi vozlišča v,
    za nek v iz množice vozlišč.
    """
    mnozica = G[vozlisce]
    novGraf = naredi_podgraf(G, mnozica)
    lokalna = neodvisnostno_stevilo(novGraf)
    return lokalna
```

4.1.3. *povprecna(G).* Ta funkcija nam vrne povprečno lokalno neodvisnost.

```
def povprecna(G):
    """
    Vrne povprečno lokalno neodvisnost grafa G
    """
    povprecje =  0
    for vozlisce in G:
        povprecje += lokalna_neodvisnost(G, vozlisce)
    povprecna_vrednost  = povprecje/ len(G)
    return povprecna_vrednost
```

4.2. **Glavna funkcija.** Glavna funkcija, s katero smo izvajali teste, je tako dobila končen izgled.

```
def preverjanje(stevilo_vozlisc):
    """
    Preveri veljavnost konjekture za vse grafe na določenem številu vozlišč.
    """
    for G in nasi_grafi(stevilo_vozlisc):
        if neodvisnostno_stevilo(G) <= 1 + povprecna(G):
            if G.hamiltonian_path() == None:
                p = G.plot()
                p.show()
                return "Ovrgli smo domnevo"
    return "Izjeme nismo ovrgli"
```

Na ta način smo oblikovali funkcije, s katerimi smo testirali grafe. Za večje število testov smo uporabili genetski algoritem.

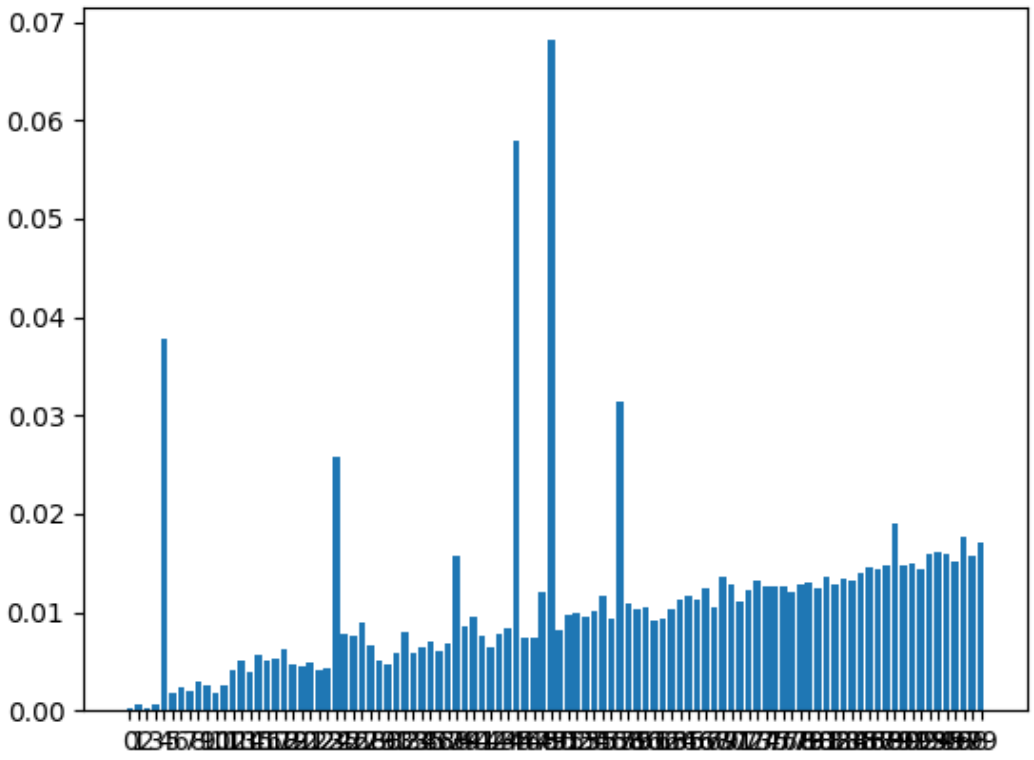
4.3. **Genetski algoritem.** Genetski algoritem je vrsta algoritma, ki temelji na Darwinovi ideji o razvoju vrst in preživetju najmočnejših. Zelo pomembno pa je, kako določimo, kdo je tu najmočnejši, zato se moramo primerno odločiti za kriterij, po katerem bomo to delali. V našem primeru smo se odločili za čim manjše neodvisnostno število.

V našem primeru sva se odločila, da bomo začetno populacijo določili s pomočjo funkcije $zacetne\_populacija(maksimalna)$, kjer določimo začetno množico povezanih grafov. Z vrednostjo maksimalna smo določili največjo velikost generacije.
Nato sva te grafe razvrstila po določenem kriteriju in določila možnosti mutacije

"populacije", torej vsakega izmed grafov. Grafu lahko dodamo ali odstranimo povezavo in dodamo ali odstranimo vozlišče, na koncu pa jih lahko še križamo.
Ko sva to naredila, sva najin genetski algoritem testirala. Test sva naredila na 10000 generacijah, ki so bile lahko velike največ 10000. Ker nam ta test ni vrnil izjeme predvidevava, da izjeme ni možno najti.

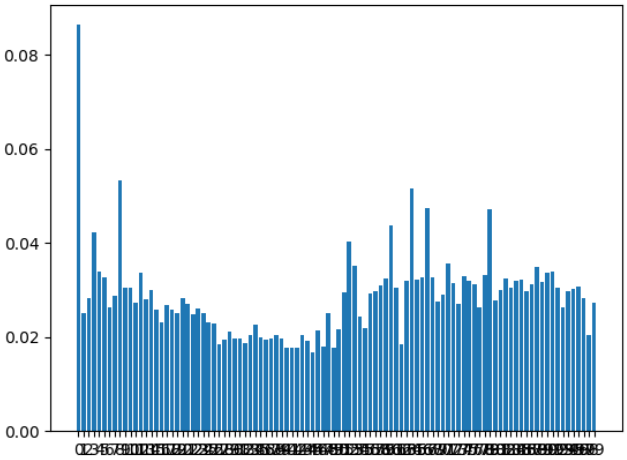
```
print(KoncniTestGA(1000000,1000000))
Domneva ni ovrzena.
```

Ker nas poleg uspešnosti kode navadno zanima tudi njena hitrost oziroma časovna zahtevnost, sva se tudi to odločila testirati. Odločila sva se za teste, ki bodo opisali odvisnost časa od števila generacij ter od največjega možnega števila osebkov v populaciji.
Najprej sva testirala odvisnost od parametra *konec*, torej števila generacij. Rezultat, ki sva ga dobila, nam je dal precej očiten rezultat.

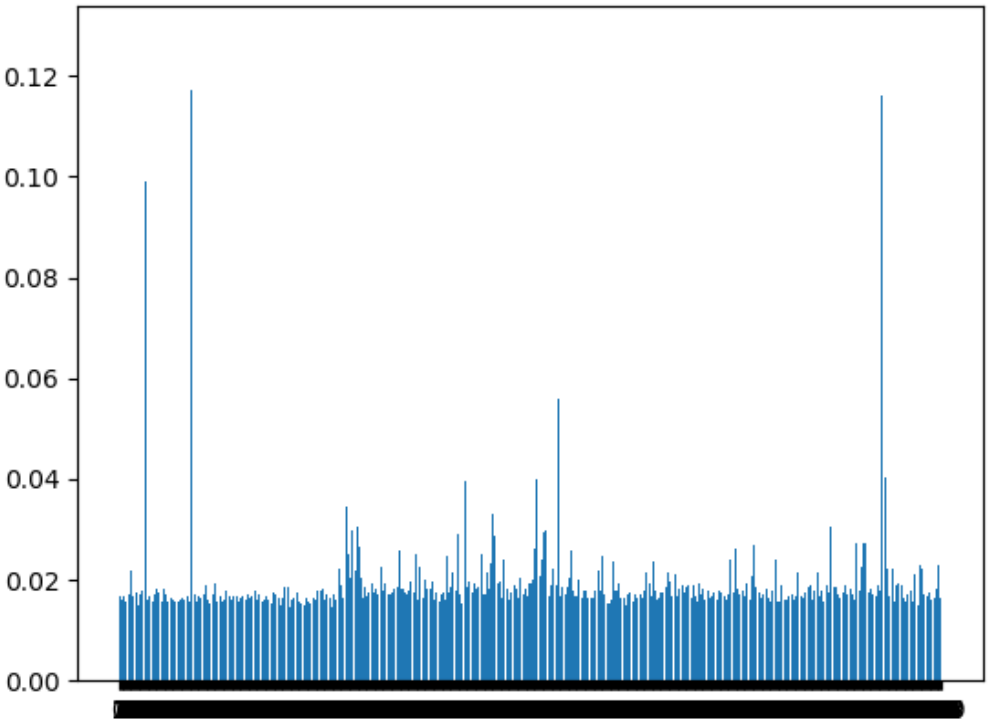


Vidimo namreč lahko, da se z rastjo števila generacij linearno povečuje tudi čas, potreben za pregled.

Bolj zanimiva je bila analiza za parameter *maksimalna*, kjer nam prvi rezultat ne da nekega enoznačnega odgovora.



Zato sva najin test spremenila tako, da sva povečala največjo možno vrednost maksimalne vrednosti, do katere sva štopala.



Tu lahko očitno vidimo, da je funkcija skoraj ni odvisna od največje dovoljene velikosti generacije.

5. Zaključek

Ob zaključku najinega dela lahko ugotoviva, da Graffitijeva Konjektura 194 najverjetneje drži, saj navkljub testu na velikem vzorcu protiprimera nisva našla.