Ekstremizacija neregularnih grafov z modifikacijo skupne σ-nepravilnosti

Tinč Arifović, Manca Kavčič

16. februar 2025

1 Uvod

V projektni nalogi sva se ukvarjala z ekstremizacijo neregularnih grafov z modifikacijo skupne σ-nepravilnosti. Projektna naloga je bila izvedena v programu SageMath in je dostopna na GitHubu, kjer so zbrane vse datoteke, ki sva jih uporabila pri samem projektu. Cilj je ugotoviti, ali je pri vseh spodaj podanih problemih največja vrednost $\sigma_t^{f(n)}(G)$ dosežena, ko je graf antiregularen.

2 Opis problema

Imamo tri probleme, za katere iščemo rešitve.

1. Naj bo $f(n) = \frac{1}{n}$. Ali je največja vrednost $\sigma_t^{f(n)}(G)$ dosežena, ko je G antiregularen graf?
2. Naj bo $f(n) = c$, kjer je c realno število iz intervala $(0, 1)$. Ali je največja vrednost $\sigma_t^{f(n)}(G)$ dosežena, ko je G antiregularen graf?
3. Naj bo $f(n)$ pozitivna funkcija, za katero velja $\lim_{n\to\infty} f(n) = 0$. Identificirajte drevesa, ki dosežejo največjo vrednost $\sigma_t^{f(n)}(G)$.

Popolna $\sigma-$nepravilnost je podana kot

$$\sigma_t(G) = \sum_{u,v\subseteq V(G)} (d_G(u) - d_G(v))^2,$$

kjer $d_G(z)$ označuje stopnjo vozlišča z v grafu G. Natančneje, definiramo indeks $\sigma_t^{f(n)}(G)$ kot

$$\sigma_t^{f(n)}(G) = \sum_{u,v\subseteq V(G)} |d_G(u) - d_G(v)|^{f(n)},$$

kjer je $n = |V(G)|, f(n)$ pa je funkcija definiran za $n \geq 4$. Najina naloga je odgovoriti na zgoraj podana vprašanja s pomočjo računalniškega testiranja. Za manjše grafe, do vključno 8 vozlišč, sva uporabila sistematičen pristop iskanja, za večje grafe pa stohastičen. Poleg tega sva se pri 2. problemu še posebej osredotočila na vrednosti zelo blizu 0 in 1.

3 Rezultati

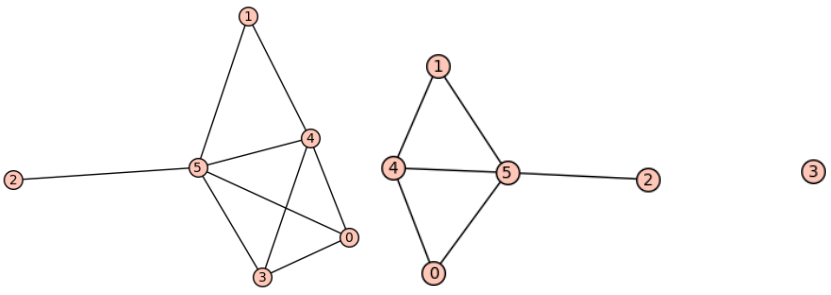
3.1 Tretji problem

Ko je $f(n) = \frac{1}{n}$, sva pri manjših grafov povsod dobila antiregularne grafe. To sva preverjala s funkcijo *je_antiregularen*, ki nama je izpisala pod grafom, če je graf antiregularen ali ne. Prav tako sva pri vseh številih vozlišč dobila kot rezultat dva grafa, torej povezan antiregularen graf skupaj s svojim komplementom, kot je prikazano spodaj na sliki za graf s 6 vozlišči. Pri iskanju grafov z največjo $\sigma_t^{f(n)}(G)$ sva uporabila *max_total_sigma_irregularity1*, ki nama je služila kot osnova za nadgradnjo pri iskanju rešitev nadaljnjih problemov.

Algorithm 1 Iskanje grafov z največjo skupno sigma nepravilnostjo

Require: n ▷ Število vozlišč
Ensure: Grafi z največjo skupno sigma nepravilnostjo
1: $grafi \leftarrow$ seznam vseh grafov na n vozliščih ▷ Ustvarimo vse grafe na n vozliščih
2: $sigme \leftarrow$ prazen seznam ▷ Seznam za vrednosti sigma nepravilnosti
3: **for** $G \in grafi$ **do**
4: Dodaj $\sigma_1(G)$ v $sigme$ ▷ Izračunamo sigma nepravilnost in shranimo vrednost
5: **end for**
6: $koncni\_grafi \leftarrow$ prazen seznam ▷ Seznam za grafe z največjo sigma nepravilnostjo
7: **for** $i \in$ indeksi največjih elementov v $sigme$ **do**
8: Dodaj $grafi[i]$ v $koncni\_grafi$ ▷ Shranjujemo grafe z največjo sigma nepravilnostjo
9: **end for**
10: Izpiši $|V| = n$ ▷ Izpis števila vozlišč
11: **for** $graf \in koncni\_grafi$ **do**
12: Prikaži $graf$ ▷ Prikažemo ustrezne grafe
13: **end for**
return None

Spodaj je prikazan primer grafov s 6 vozlišči. Levi ima stopnje vozlišč $(1, 2, 3, 3, 4, 5)$, desni pa $(0, 1, 2, 2, 3, 4)$.



Slika 1: *Grafa s 6 vozlišči*

Za večje grafe sva s pomočjo funkcije *regularni_grafi* generirala regularne

grafe, s funkcijo *generate_antiregular_graph* generirala antiregularna grafa in s pomočjo funkcije *nakljucni_grafi* generirala naključne grafe. Pri tem pa sva skupno sigma nepravilnost računala za regularne, antiregularne in 5000 naključnih grafov.

Algorithm 2 Generiranje regularnih grafov

Require: n ▷ Število vozlišč
Ensure: Seznam vseh regularnih grafov na n vozliščih
regularni_grafi ← prazen seznam ▷ Inicializiramo seznam za shranjevanje regularnih grafov
for $i \leftarrow 2$ to $n-1$ **do**
    if $n \cdot i$ ni sodo število **then** ▷ Za obstoj i-regularnega grafa mora biti $n \cdot i$ sodo
        continue ▷ Preskočimo to vrednost i
    else
        niz ← niz ukazov za nauty geng ▷ Oblikujemo niz za generiranje regularnih grafov
        for $G \in$ graphs.nauty_geng(*niz*) **do**
            Dodaj G v *regularni_grafi* ▷ Dodamo graf v seznam
        end for
    end if
end for
    return *regularni_grafi*

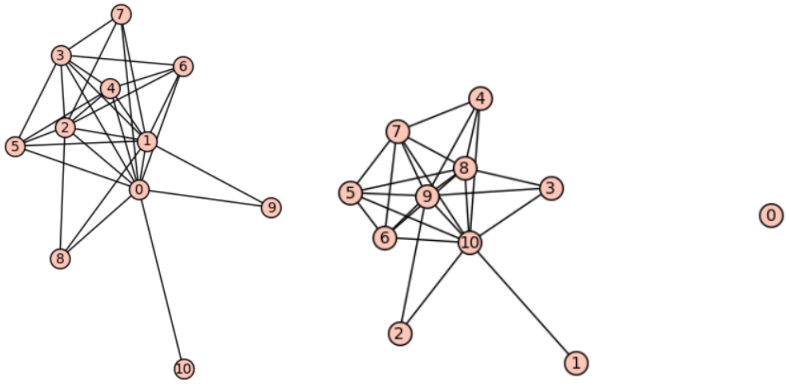
Algorithm 3 Generiranje antiregularnih grafov

function AntiregularniGrafi(n)
    $G \leftarrow$ prazen graf z n vozlišči
    antiregularni_grafi ← prazen seznam
    vozlisca ← $[0, 1, \ldots, n-1]$
    for $i \leftarrow 1$ to $n-1$ **do**
        Dodaj povezavo $(vozlisca[0], vozlisca[i])$
    end for
    for $i \leftarrow 1$ to $n-2$ **do**
        for $j \leftarrow i+1$ to $n-i-1$ **do**
            Dodaj povezavo $(vozlisca[i], vozlisca[j])$
        end for
    end for
    Dodaj G v *antiregularni_grafi*
    Dodaj $G.complement()$ v *antiregularni_grafi*
    return *antiregularni_grafi*
end function

Tudi pri večjih grafih sva dobila povsod antiregularne grafe kot rezultat.

Algorithm 4 Generiranje naključnih grafov

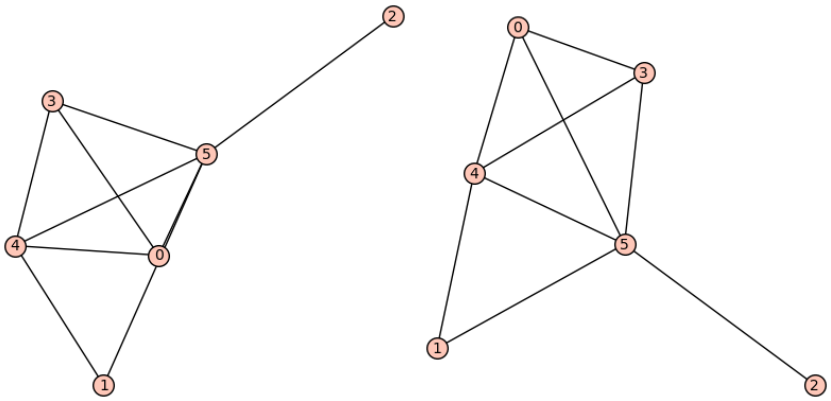
Require: n, st ▷ Število vozlišč in število grafov za generiranje
Ensure: Seznam naključnih grafov
1: $seznam\_grafov \leftarrow$ prazen seznam ▷ Inicializiramo seznam za shranjevanje grafov
2: **for** $i \leftarrow 1$ to st **do**
3: $mozne\_povezave \leftarrow$ vse možne povezave med pari vozlišč ▷ Ustvarimo seznam vseh možnih povezav
4: $izbrane\_povezave \leftarrow$ naključno izbran podmnožica iz $mozne\_povezave$ ▷ Naključno izberemo povezave
5: $G \leftarrow$ prazen graf na n vozliščih ▷ Ustvarimo prazen graf
6: Dodaj povezave $izbrane\_povezave$ v G ▷ Dodamo izbrane povezave v graf
7: Dodaj G v $seznam\_grafov$ ▷ Dodamo graf v seznam
8: **end for**
return $seznam\_grafov$



Slika 2: *Antiregularna grafa z 11 vozlišči*

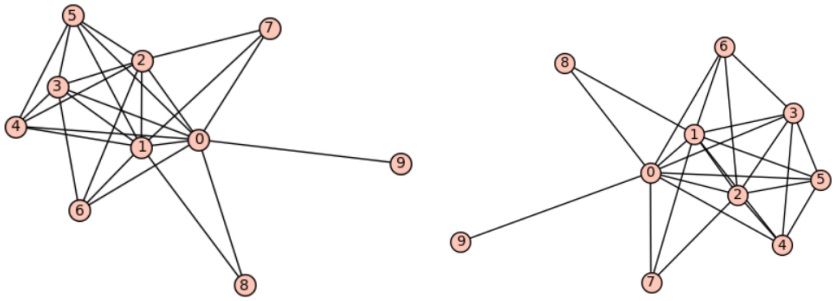
3.2 Četrti problem

Pri drugem problemu sva pri manjših grafih za zelo majhne c blizu 0 in blizu 1 dobivala enake grafe kot pri prvem problemu torej antiregularne grafe, torej povezan antiregularen graf in njegov komplement, in prav tako za vse vmesne vrednosti c. Za c sva uporabila vrednosti $\{\frac{1}{1000}, \frac{1}{100}, \frac{1}{10}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{9}{10}, \frac{99}{100}, \frac{999}{1000}\}$.



Slika 3: *Grafa s 6 vozlišči za* $c = \frac{1}{1000}$ *(levo) in* $c = \frac{99}{100}$ *(desno)*

Pri večjih grafih sva prav tako dobila povsod antiregularne grafe. Spodaj je podan primer grafa z 10 vozlišči (pri čemer sva kot rezultat dobila tudi njuna komplementa, ki pa nista na sliki).



Slika 4: *Grafa z 10 vozlišči za* $c = \frac{1}{100}$ *(levo) in* $c = \frac{99}{100}$ *(desno)*

3.3 Peti problem

Za ta problem sva se odločila uporabiti 9 funkcij, za katere velja

$$\lim_{n\to\infty} f(n) = 0.$$

Funkcije so razvrščene glede na hitrost njihovega približevanja vrednosti 0, od najhitreje padajočih do najpočasneje padajočih:

1. e^{-x^2}
2. $\frac{e^{-x}}{\sqrt{x}}$
3. e^{-x}
4. $\frac{1}{x^2}$
5. $e^{-\sqrt{x}}$
6. $\frac{|\cos(x)|}{x}$
7. $\frac{\sin^2(x)}{x}$
8. $\frac{1}{\ln^2(x)}$
9. $\frac{1}{\sqrt{x}}$

Pri tem delu sva najprej definirala funkcijo *drevesa* in s pomočjo *TreeIterator* generirala drevesa, na katerih sva potem uporabila funkcijo

max_total_sigma_irregularity3_drevesa. Spodnja tabela za vsako funkcijo prikazuje, kakšno porazdelitev stopenj ima posamezno vozlišče pri določenem vozlišču. Torej za drevo na 5 vozliščih imamo povsod 3 vozlišča stopnje 1, 1 vozlišče stopnje 2 in 1 vozlišče stopnje 3. Poleg tega pa so z rdečo obarvane vse porazdelitve, ki so drugačne od ostalih in so v manjšini. Po večini se to zgodi pri funkcijah, ki počasneje padajo proti 0.

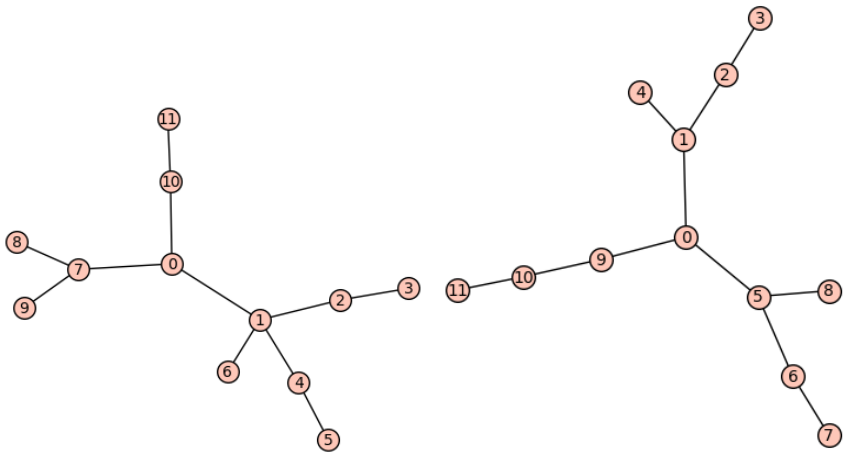
f(x) \ Št. vozlišč	4	5	6	7	8
e^{-x^2}	1:2, 2:1	1:3, 2:1, 3:1	1:3, 2:2, 3:1	1:3, 2:3, 3:1	1:4, 2:2, 3:2
$\frac{e^{-x}}{\sqrt{x}}$	1:2, 2:1	1:3, 2:1, 3:1	1:3, 2:2, 3:1	1:3, 2:3, 3:1	1:4, 2:2, 3:2
⋮	⋮	⋮	⋮	⋮	⋮
$\frac{\sin^2(x)}{x}$	1:2, 2:1	1:3, 2:1, 3:1	1:3, 2:2, 3:1	1:3, 2:3, 3:1	1:4, 2:2, 3:2
$\frac{1}{\ln^2(x)}$	1:3, 3:1	1:3, 2:1, 3:1	1:3, 2:2, 3:1	1:4, 2:2, 4:1	1:4, 2:2, 3:2
$\frac{1}{\sqrt{x}}$	1:3, 3:1	1:3, 2:1, 3:1	1:3, 2:2, 3:1	1:4, 2:2, 4:1	1:4, 2:2, 3:2

9	10	11	12	13
1:4, 2:3, 3:2	1:4, 2:4, 3:2	1:5, 2:4, 3:1, 4:1	1:5, 2:4, 3:3	1:6, 2:4, 3:2, 4:1
1:4, 2:3, 3:2	1:5, 2:3, 3:1, 4:1	1:5, 2:4, 3:1, 4:1	1:6, 2:3, 3:2, 4:1	1:6, 2:4, 3:2, 4:1
⋮	⋮	⋮	⋮	⋮
1:4, 2:3, 3:2	1:5, 2:3, 3:1, 4:1	1:5, 2:4, 3:1, 4:1	1:6, 2:3, 3:2, 4:1	1:6, 2:4, 3:2, 4:1
1:5, 2:2, 3:1, 4:1	1:5, 2:3, 3:1, 4:1	1:5, 2:4, 3:1, 4:1	1:6, 2:3, 3:2, 4:1	1:6, 2:4, 3:2, 4:1
1:5, 2:2, 3:1, 4:1	1:5, 2:3, 3:1, 4:1	1:6, 2:2, 3:2, 4:1	1:6, 2:3, 3:2, 4:1	1:6, 2:4, 3:2, 4:1

14	15	16	17
1:6, 2:5, 3:2, 4:1	1:7, 2:4, 3:3, 4:1	1:7, 2:5, 3:3, 4:1	1:7, 2:6, 3:3, 4:1
1:6, 2:5, 3:2, 4:1	1:7, 2:4, 3:3, 4:1	1:7, 2:5, 3:3, 4:1	1:7, 2:6, 3:3, 4:1
⋮	⋮	⋮	⋮
1:6, 2:5, 3:2, 4:1	1:7, 2:4, 3:3, 4:1	1:7, 2:5, 3:3, 4:1	1:7, 2:6, 3:3, 4:11
1:6, 2:5, 3:2, 4:1	1:7, 2:4, 3:3, 4:1	1:7, 2:5, 3:3, 4:1	1:8, 2:5, 3:2, 4:2
1:7, 2:3, 3:3, 4:1	1:7, 2:4, 3:3, 4:1	1:8, 2:4, 3:2, 4:2	1:8, 2:5, 3:2, 4:2

Tabela 1: Tabela, ki za posamezno število vozlišč in določeno funkcijo prikazuje koliko je vozlišč določene stopnje na tem drevesu (*stopnja vozlišča:št. vozlišč*)

Pri tem sva opazila, da se vozlišče stopnje 3 pojavi, že pri drevesih s 4 vozlišči, in se ohrani povsod naprej, medtem, ko se vozlišče stopnje 4 prvič pojavi pri drevesih s 7 vozlišči, vendar pa se ta ohrani, šele pri drevesih, ki imajo vsaj 9 vozlišč. V tabelo sva zapisala porazdelitev vozlišč vse do dreves z največ 17 vozlišči in pri tem dobila, da je najvišja stopnja posameznega vozlišča 4, kar se ni spremenilo vse do dreves z 20 vozlišči. Poleg tega pa so porazdelitve, ki odstopajo od večine ponekod nekako podobne porazdelitvam pri drevesih z enim številom vozlišč več, torej odstopanja pri drevesih z 9 vozliči (obarvano z rdečo) so podobna porazdelitvam pri 10 vozliščih.



Slika 5: *Drevesi z 12 vozlišči ter razporeditvijo 1:6, 2:3, 3:2, 4:1 (levo) in 1:5, 2:4, 3:3 (desno)*

4 Zaključek

Kot komentar bi dodala, da se je čas izvajanja programa povečeval s povečevanjem števila vozlišč, saj se z večjim številom vozlišč poveča število različnih grafov, obenem pa je za vsak graf pri izračunu $\sigma_t^{f(n)}(G)$ potrebnih več računskih operacij. Pri drugem problemu je bilo za c zelo blizu 0 in 1 potrebnega več časa za izračun vrednosti kot pri ostalih vrednostih c.