Poročilo Projekta 2025

Število geodetskih podpoti

Mentorja: Riste Škrekovski, Timotej Hrga
Nina Smole, Martin Čadež

UVOD

Namen raziskave je bil sistematično preučiti ekstremalne grafe glede na število geodetskih podpoti (angl. *geodesic path number*, okrajšano *gpn*) za različne strukturne razrede povezanih grafov. Specifično nas je zanimala identifikacija grafov, ki v danem razredu dosežejo maksimalno možno vrednost *gpn* za fiksno število vozlišč.

Reševanja problema smo se lotili v treh glavnih korakih. V prvem koraku smo izvedli analizo števila geodetskih podpoti vseh enostavnih povezanih grafov z manj kot desetimi vozlišči. V drugem koraku smo na podlagi analize postavili hipotezo o tem, kateri grafi so optimalni za merilo dano *gpn*. V zadnjem koraku smo za grafe z večjim število vozlišč postavljeno hipotezo preverili z uporabo hevrističnih metod.

TEORETIČNO OZADJE

Za natančno obravnavo problema si najprej poglejmo naslednje definicije:

Definicija **Pot grafa**

Naj bo G povezan, enostaven graf. Pot dolžine ℓ, je zaporedje vozlišč $(v_0, v_1, \ldots, v_\ell)$ brez ponovitev, kjer je vsak par zaporednih vozlišč povezan s povezavo grafa G, tj:

$$\forall\, i \in \{1, 2, \ldots, \ell\} : v_{i-1}v_i \in E(G)$$

Trivialna pot dolžine 0 vsebuje eno vozlišče.

Definicija **Geodetska pot grafa**

Naj bo G povezan, enostaven graf, in naj bosta $u, v \in V(G)$ dve vozlišči grafa. Geodetska pot med u in v je pot $(v_0, v_1, \ldots, v_\ell)$, ki ima minimalno možno dolžino med vsemi potmi, ki povezujejo u in v.

Dolžino geodetske poti označimo z $d_G(u, v)$ in jo imenujemo *razdalja* med u in v.

Definicija **Število geodetskih podpoti**

Naj bo G povezan, enostaven graf. Število geodetskih podpoti, označeno $gpn(G)$, je definirano kot število vseh geodetskih (najkrajših) poti v grafu, vključno s trivialnimi potmi:

$$\mathrm{gpn}(G) = \sum_{u,v \in V(G)} \text{število geodetskih poti med } u \text{ in } v$$

Torej za vsak par različnih vozlišč povezanega grafa obstaja vsaj ena geodetska pot, ter za vsako vozlišče obstaja natanko ena trivialna pot.

V raziskavi smo se osredotočili na naslednje strukturne razrede povezanih grafov:

Definicija **Dvodelni graf**

Graf $G = (V, E)$ je *dvodelen*, če obstaja razcep (particija) množice vozlišč $V = A \cup B$, $A \cap B = \emptyset$, tako da za vsako povezavo $uv \in E(G)$ velja:

$$\forall u, v \in V(G) : (u \in A \wedge v \in B) \vee (u \in B \wedge v \in A)$$

Torej, vse povezave povezujejo vozlišča iz različnih particij.

Definicija **Graf brez trikotnikov**

Graf G je *brez trikotnikov* (angl. triangle-free graph), če v njem ne obstaja cikel dolžine 3, torej če graf G ne vsebuje podgrafa, izomorfnega ciklu C_3.

Trditev **Dvodelni grafi so brez trikotnikov**

Vsak dvodelni graf je hkrati graf brez trikotnikov. Obratno ne velja.

Definicija **Kubičen graf**

Graf $G(V, E)$ je *kubičen* (3-regularen), če ima vsako vozlišče stopnjo 3:

$$\forall\, v \in V(G) : deg(G) = 3$$

Torej vsako vozlišče je sosednje natanko trem vozliščem. Število povezav je enako:

$$|E(G)| = \frac{3|V(G)|}{2}$$

Trditev **Obstoj kubičnih grafov**

Kubični grafi lahko obstajajo le na sodem številu vozlišč.

Definicija **Ekstremalen graf**

Naj bo $\mathcal{G}$ razred grafov in $P(G)$ neka invarianta (neodvisna lastnost grafa). Graf G^* je ekstremalen glede na P, če velja:

$$G^* = \arg\max_{G \in \mathcal{G}} P(G)$$

ko želimo maksimizirati P oziroma:

$$G^* = \arg\min_{G \in \mathcal{G}} P(G)$$

ko želimo minimizirati P.

DEFINICIJA PROBLEMA

V naši študiji smo iskali grafe, ki maksimizirajo invarianto $gpn(G)$ znotraj razreda enostavnih povezanih označenih (angl. *labeled*) grafov na fiksnem številu vozlišč. Slednji problem lahko definiramo z naslednjim celoštevilskim linearnim programom:

METODOLOGIJA REŠEVANJA PROBLEMA

Problema smo se lotili na sledeč način:

Razvojno okolje (angl. *development environment*):
Ker je iskalni prostor (angl. search space) vseh enostavnih povezanih označenih grafov izjemno velik, število takih grafov za naraščajoče število vozlišč sledi zaporedju:

$$1,\ 1,\ 2,\ 6,\ 21,\ 112,\ 853,\ 11117,\ 261080,\ 11716571,\ \ldots$$

smo se naloge seveda lotili s pomočjo programskih jezikov in računalniške moči. Zaradi številnih sofisticiranih implementacij programske opreme za delo z grafi, smo se odločili, da uporabimo razvojno okolje **SageMath**. Slednjega smo vspostavili s sistemom Docker, saj je namestitev neposredno na določene operacijske sisteme kot je Windows brez WSL-ja (Windows Subsystem for Linux) nepraktična. Torej, s pomočjo Docker-ja smo vzpostavili virtualni vsebnik (angl. container), v katerem smo izvedli celotno analizo. Z virtualizacijo smo si zagotivili izolacijo okolja od gostiteljevega operacijskega sistema in enotno delovno okolje na različnih platformah. Artifakt Docker-jeve slike lahko najdete *tukaj*.

Generacija podatkov:
V prvem eksperimentalnem koraku smo definirati objekte iskalnega razreda. Te smo za manjše grafe lokalno generirali sami, za večje grafe pa smo se poslužili podatkovne zbirke *Bruce D. McKay za grafe*, katere repozitorij je dostopen *tukaj*.

V večji meri smo uporabili SageMath-ov modul *nauty*, natančneje vmesnik *nauty_geng* programa **geng**, za učinkovito generiranje neizomorfnih grafov z določenimi lastnostmi. V prvotni fazi smo se osredotočili predvsem na tri strukturne razrede: kubične grafe, dvodelne grafe in grafe brez trikotnikov. Za vsakega izmed teh razredov smo v programskem jeziku Python implementirali ustrezne razrede (*OOP class*), ki omogočajo generiranje vseh možnih grafov na determinističnem številu vozlišč. Omembe vredno je, da vsak razred vključuje lastnosti grafov ter možnost vizualizacije posameznega objekta iz razreda. Za enostavnejšo implementacijo smo uporabili knjižnici *networkx* in *matplotlib*. Izvorna koda se nahaja *tukaj*.

Podrobneje, lokalno smo generirali vse neizomorfne enostavne povezane označene grafe na do vključno devetih vozliščih. Za grafe z desetimi vozlišči, smo zaradi omejenih računalniških resursov, podatke naložili. Pri zgoraj omenjenih specifičnih strukturnih razredih, nismo imeli težav z generacijo, saj so te skupine bistveno manjše (za grafe $|V(G)| \geq 10$). Izvorna koda je dostopna *tukaj*.

Vsak graf smo enkodirali z **graph6** identifikacijskim ključem, saj ta enolično določa vsak enostaven graf znotraj našega iskalnega prostora.

Izračun invariante:
Algoritem za izračun števila geodetskih podpoti je priložen med prilogami, izvorna koda pa se nahaja *tukaj*.

Podatke smo shranili v datoteke formata `CSV` (Comma Separated Values), ki smo jih po potrebi stisnili zaradi omejitev velikosti datotek na GitHub-u. Pri tem smo uporabili poznano knjižnico za delo s podatki *pandas* ter interaktivno računalniško okolje *Jupyter Notebook*.

Za olajšan pristop do podatkovnih zbirk so spodaj priložene povezave:

- *link*: zbirka vseh enostavnih povezanih grafov na do vključno devetih vozliščih
- *link*: zbirka vseh enostavnih povezanih grafov na desetih vozliščih
- *link*: zbirka vseh grafov iz zgoraj omenjenih specifičnih strukturnih razredov
- *link*: zbirka vseh dvodelnih grafov na 11 vozliščih
- *link*: zbirka vseh dvodelnih grafov na 12 vozliščih

Prečno iskanje (angl. *exhaustive search*):
Na podlagi prvih treh zgoraj opisanih podatkovnih zbirk smo iz razredov izločili zgolj *ekstremalne grafe* glede na invarianto *gpn*. Pri obdelavi in filtriranju podatkov smo ponovno uporabili knjižnico *pandas* ter sodobnejšo knjižnico *polars*, razvito v okviru rustovega ekosistema. Glavna prednost slednje je podpora lenega nalaganja (angl. *lazy loading*), ki omogoča učinkovito filtriranje velikih CSV-datotek brez nepotrebne porabe virov.

V spodnji tabeli so zbrani ekstremalni grafi za posamezen razred glede na število vozlišč:

graph6 ID	$\|V(G)\|$	$\|E(G)\|$	brez trikotnikov	dvodelen	kubičen	$gpn(G)$
@	1	0	ja	ja	ne	1
*A_*	2	1	ja	ja	ne	3
BW	3	2	ja	ja	ne	6
Bw	3	3	ne	ne	ne	6
C]	4	4	ja	ja	ne	12
DFw	5	6	ja	ja	ne	20
*EFz_*	6	9	ja	ja	ja	33
*F?zv_*	7	11	ja	ja	ne	49
*F?v_*	7	12	ja	ja	ne	49
*G?zVf_*	8	14	ja	ja	ne	74
*G?zvf_*	8	15	ja	ja	ne	74
H?BvUrw	9	17	ja	ja	ne	105
H?BvVrw	9	18	ja	ja	ne	105
H?ovfbo	9	16	ja	ja	ne	105
I?BvUqw}?	10	21	ja	ja	ne	151

Prikazi zgornjih grafov so priloženi v prilogi. Na podlagi tabele lahko podamo hipotezo:

Hipoteza **Ekstremalni grafi za** *gpn*

Za razred enostavnih povezanih grafov nad n-timi vozlišči so ekstremalni grafi dvodelni.

Slednjo lahko potrdimo že z diagramoma v prilogi, ki sta bila ustvarjena v okolju programskega jezika R z uporabo paketov iz skupine *tidyverse* in paketa *ggplot2* za vizualizacije. Izvorna koda se nahaja *tukaj*.

Na podlagi hipoteze, smo prečno preiskali (angl. *brute-force search*) vse dvodelne grafe na enajstih in dvanajstih vozliščih in dobili naslednje rezultate o ekstremalnih grafih:

graph6 ID	$\|V(G)\|$	$\|E(G)\|$	brez trikotnikov	dvodelen	kubičen	$gpn(G)$
J??E@w{}Fo?	11	20	ja	ja	ne	209
K??E@w{}Foˆ?	12	25	ja	ja	ne	303

Z dodatnim opazovanjem strukture optimalnih grafov, si lahko še dodatno skrčimo iskalni prostor, saj opazimo da so grafi sistematično dvodelni:

Hipoteza **Ekstremalni grafi za** *gpn*

Največje število geodetskih podpoti za razred enostavnih povezanih grafov nad n-timi vozlišči dosežejo **uravnoteženi dvodelni** in precej regularni grafi, kar pomeni da je particija vozlišč čim bolj enakomerna:

- n je sod $\implies$ optimalna particija: (k, k)
- n je lih $\implies$ optimalna particija: $(k, k+1)$

Hevristične metode iskanja:
Na podlagi zgornje hipoteze smo implementirali razred, ki generira naključen uravnotežen dvodelen graf, pri katerem lahko stohastično uravnavamo pričakovano gostoto povezav. Izvorna koda se nahaja *tukaj*.

Ker se je iskalni prostor znatno zmanjšal, smo za iskanje ekstremalnih večjih grafov od 11 do vključno 30 vozlišči uporabili metodo naključnega iskanja (angl. *random search*). Izvorna koda je dostopna *tukaj*. Hevristični rezultati se nahajajo *tukaj*.

Za nadaljnjo optimizacijo smo uporabili knjižnico *simanneal*, ki poenostavi implementacijo simuliranega žarjenja (angl. *simulated annealing, SA*). Začetno stanje smo inicializirali z optimalnim grafom iz prejšnjega koraka. Vsako naslednje stanje smo dobili z majhno perturbacijo grafa, kjer smo z enakomerno porazdeljeno verjetnostjo odstranili, dodali ali spremenili povezavo. Izboljšani rezultati so dostopni *tukaj*.

Testiranje hipoteze:
Hipotezo smo preverili s Studentovim t-testom za dva neodvisna vzorca (angl. *Student's two-sample t-test*) s pomočjo knjižnice *scipy* z uporabo modula *stats*. Za prvi vzorec smo izbrali grafe iz populacije vseh enostavnih povezanih grafov, za drugi pa le grafe, ki ustrezajo razredu, opisanemu v hipotezi. Ničelna hipoteza je predpostavljala, da je povprečje *gpn* vrednosti uravnoteženih dvodelnih grafov večje od povprečja *gpn* vrednosti grafov iz splošne populacije. Z uporabo 5-odstotne stopnje napake smo hipotezo potrdili. Izvorna koda je na voljo *tukaj*.

Priloga z psevdoimplementacijo funkcije gpn:

Algoritem za izračun števila geodezičnih poti:

Vhod: Povezan enostaven neutežen graf G, logična spremenljivka `count_trivial`
Izhod: naravno število geodetskih poti grafa G

Časovna zahtevnost: $\mathcal{O}(|V|(|V| + |E|))$
Prostorska zahtevnost: $\mathcal{O}(|V| + |E|)$

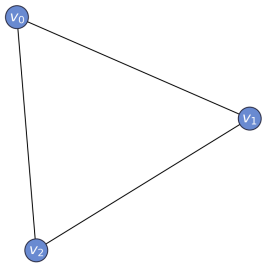
```
function GPN(G = (V, E), count_trivial)
    if count_trivial then
        T ← |V|
    else
        T ← 0
    foreach s ∈ V do
        foreach v ∈ V do
            // najkrajša razdalja od s do v
            d_G(s, v) ← ∞
            // število najkrajših poti od s do v
            σ_s(v) ← 0
        d_G(s, s) ← 0
        σ_s(s) ← 1
        // FIFO vrsta (queue)
        Q ← ∅
        enqueue(Q, s)
        // BFS
        while Q ≠ ∅ do
            u ← dequeue(Q)
            foreach w ∈ Adj(u) do
                // prvič odkrijemo vozlišče w
                if d_G(s, w) = ∞ then
                    d_G(s, w) ← d_G(s, u) + 1
                    σ_s(w) ← σ_s(u)
                    enqueue(Q, w)
                // w smo že obiskali na najkrajši razdalji
                if d_G(s, w) = d_G(s, u) + 1 then
                    σ_s(w) ← σ_s(w) + σ_s(u)
        // preštejemo vse neurejene pare vozlišča u
        foreach t ∈ V : t > s do
            T ← T + σ_s(t)
    return T
```

Priloga z ekstremalnimi grafi:

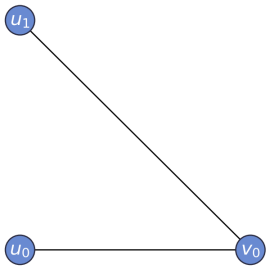
graph6 encoding label: @

graph6 encoding label: A_

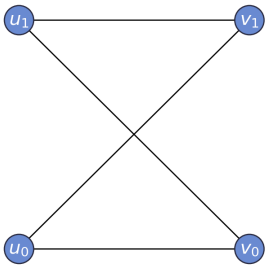
graph6 encoding label: Bw



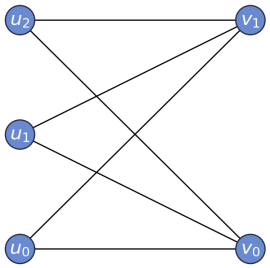
graph6 encoding label: BW



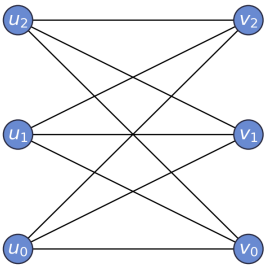
graph6 encoding label: C]



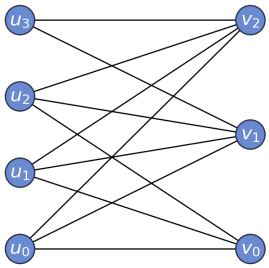
graph6 encoding label: DFw



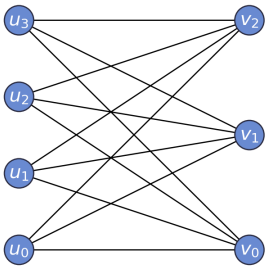
graph6 encoding label: EFz_



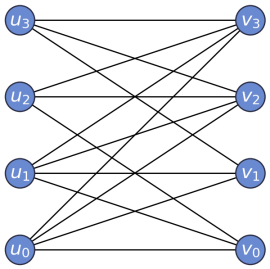
graph6 encoding label: F?zv_



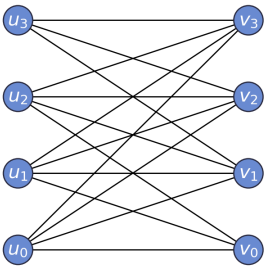
graph6 encoding label: F?~v_



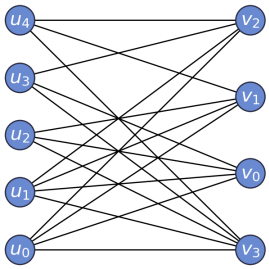
graph6 encoding label: G?zVf_



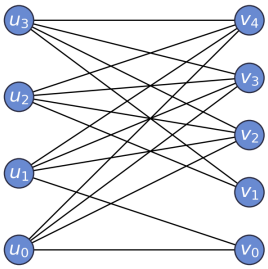
graph6 encoding label: G?zvf_



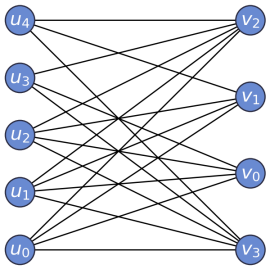
graph6 encoding label: H?BvUrw



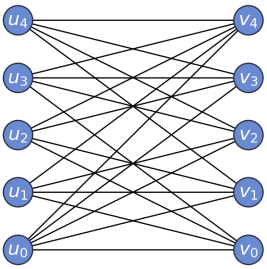
graph6 encoding label: H?ovfbo



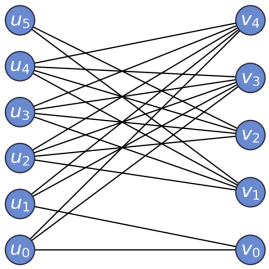
graph6 encoding label: H?BvVrw



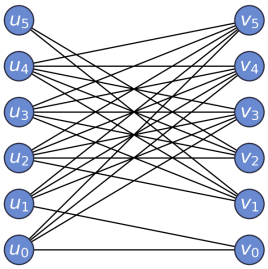
graph6 encoding label: I?BvUqw}?



graph6 encoding label: J??E@w{}Fo?

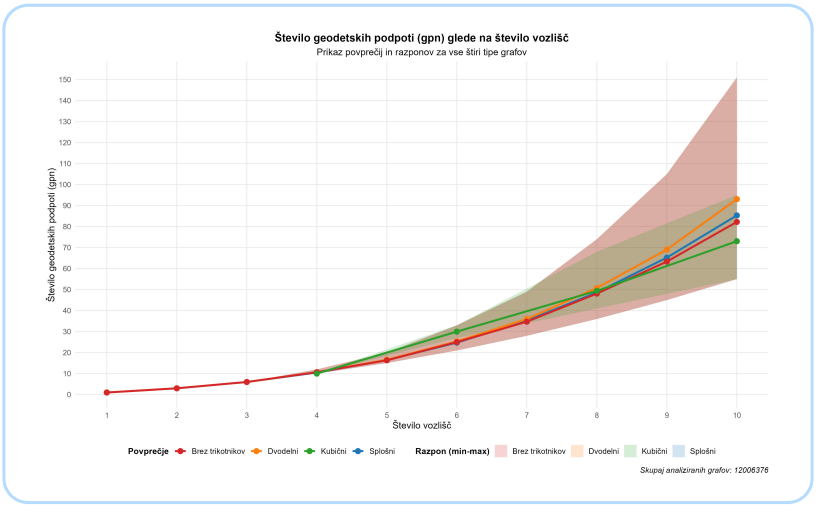


graph6 encoding label: K??E@w{}Fo^?

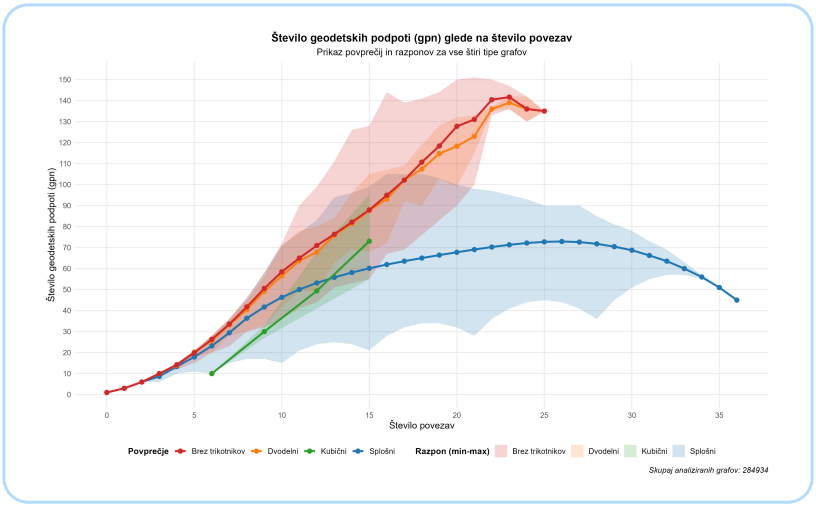


Priloga z diagrami statistične analize:

Za vse obravnavane razrede grafov smo opazili monotono naraščanje števila geodetskih podpoti s številom vozlišč, kar je dobro razvidno na črtnem diagramu:



Na podlagi analize povprečnih vrednosti so se dvodelni grafi izkazali kot optimalni razred, sledili so jim splošni grafi, medtem ko so kubični grafi dosegali najnižje vrednosti *gpn* pri enakem številu vozlišč. Ker je invarianta *gpn* precej odvisna od števila povezav že iz same definicije, smo v naslednjem diagramu prikazali glede na število povezav. Poudariti je treba metodološki vidik analize, saj predstavljeni diagram prikazuje povprečne vrednosti *gpn* glede na število povezav za grafe, ki so bili generirani odvisnosti od števila vozlišč. Torej tukaj nismo obravnavali vseh možnih grafov na m-tih povezavah, temveč zgolj grafe z danim številom vozlišč, razvrščenih glede na število povezav.



Iz diagrama je razvidno, da se povprečna vrednost *gpn* z večanjem števila povezav najprej povečuje, doseže maksimum, nato pa začne upadati za splošen razred. Na začetku dodatne povezave ustvarjajo alternativne geodetske poti, kar vrednost *gpn* povečuje. Vendar ko graf doseže preveliko gostoto povezav in se približa polnemu grafu, večina razdalj med vozlišči postane dolžine ena, kar posledično zmanjša število različnih najkrajših poti. Polni graf K_n, podobno kot drevesa, dosega minimalno vrednost *gpn*, tj. $gpn(K_n) = \binom{n}{2} + n$.